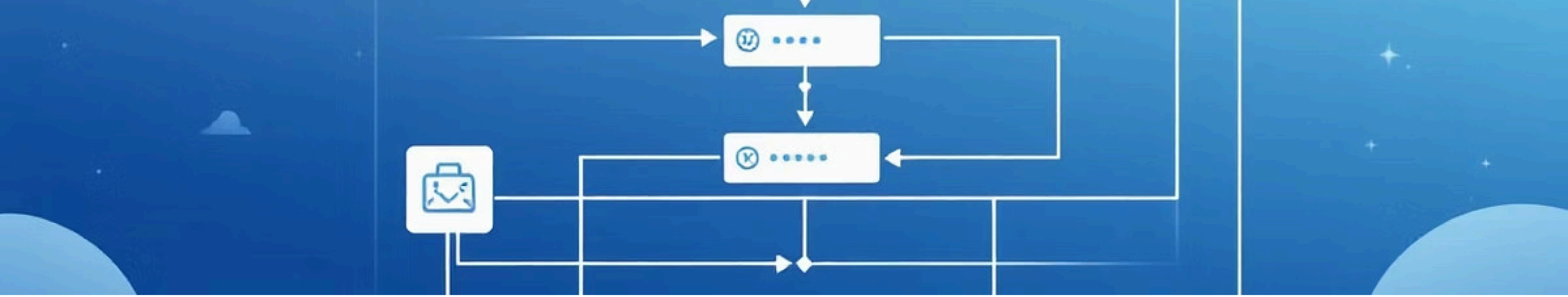
Mastering Ansible Inventory & Host Management

A comprehensive guide to understanding hosts, inventory configuration, and command-line tools for effective automation management in enterprise environments.

GitHub: https://github.com/SupawitSaelim



Understanding the Architecture

Control Host

Where Ansible CLI is executed - typically your local machine or dedicated management server

Managed Hosts

Target devices (servers, network appliances, or any computer) you configure with Ansible automation

Inventory File

Defines all managed hosts and groups for running automation tasks efficiently

Reference: github.com/SupawitSaelim

Inventory File Essentials

Default Location

The default location for Ansible inventory is `/etc/ansible/hosts`. Setting hosts globally eliminates the need to specify inventory file location using the `-i` option.

Supported Formats

Inventory files are written in either INI (Initialisation file) or YAML format, providing flexibility for different team preferences.

Reference: SupawitSaelim on GitHub

Creating Host Groups

1

INI Format Groups

```
[webservers]
192.168.1.92
web1.example.com

[dbservers]
192.168.1.93
```

Create new INI sections with group names in square brackets

2

YAML Format Groups

```
webservers:
  hosts:
    192.168.1.92:
    web1.example.com:
dbservers:
  hosts:
    192.168.1.93:
```

YAML provides a more structured, readable approach to inventory management

Remember: Variable names cannot begin with a number (Python requirement)

Reference: [GitHub Profile]()

Managing Inventory Variables

Group Variables

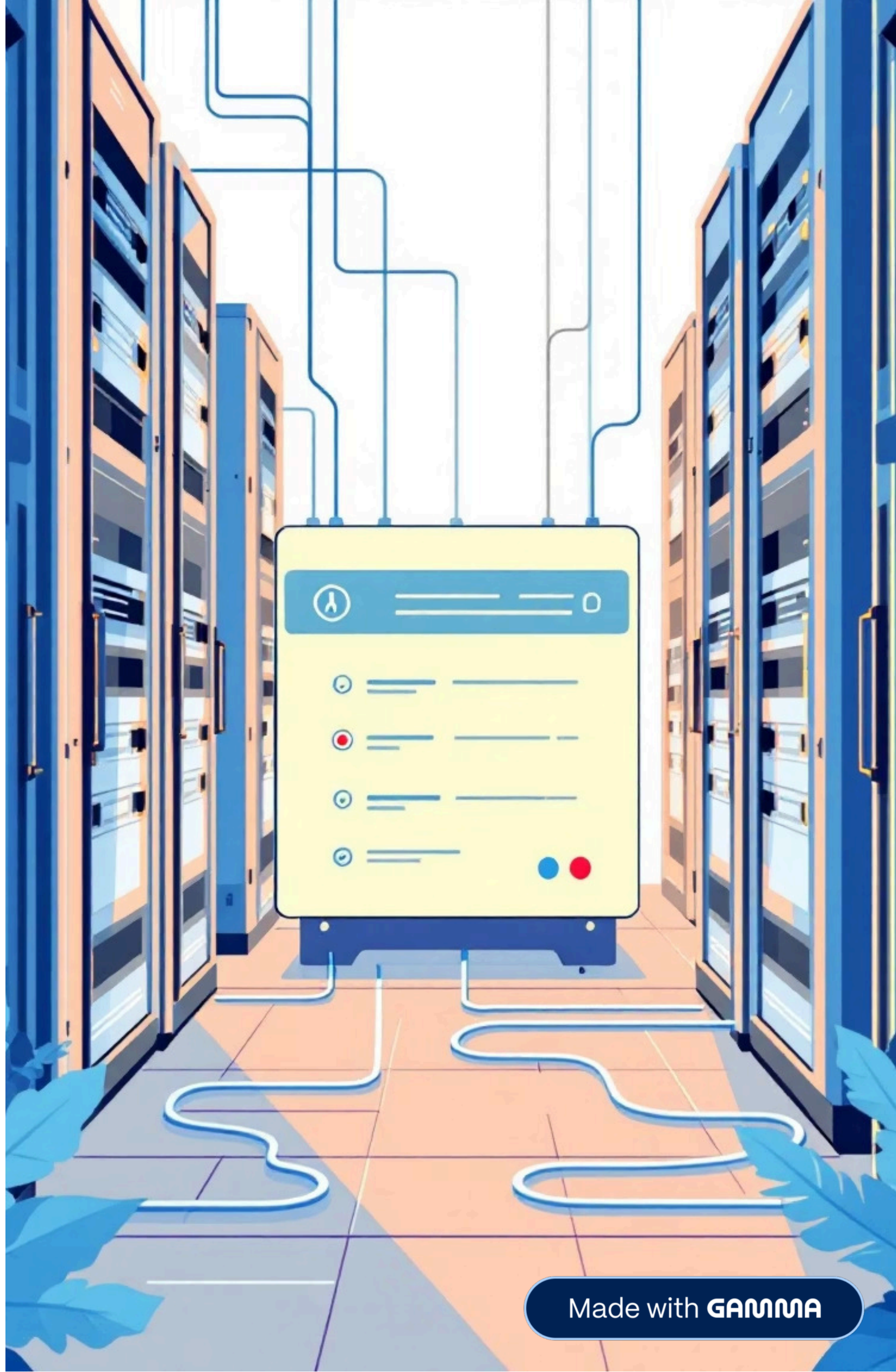
Define variables for multiple hosts using the special `:vars` section format:

```
[dbservers]
db1.example.com
db2.example.com

[dbservers:vars]
ansible_user=admin
ansible_port=2200
```

This approach ensures consistent configuration across all hosts in a group, reducing maintenance overhead and configuration drift.

GitHub Profile



Advanced Inventory Options

Multiple Inventory Files

Create directories with multiple inventory files for better organisation and environment separation

Dynamic Inventory

Pull inventory dynamically from cloud providers or external systems for real-time host discovery

Mixed Approaches

Combine static and dynamic inventory files using multiple `-i` arguments for maximum flexibility

Reference: github.com/SupawitSaelim

Essential CLI Tools

Basic Module Execution

Use the -m argument to specify Ansible modules:

```
ansible all -i inventory -m ping
```

This command runs the ping module on all hosts defined in your inventory file.

Ansible Playbook

Playbooks define what to do on which devices. Essential arguments include:

- -K: Ask for privilege escalation password
- --check: Run in check mode
- --diff: Show file differences
- -v/-vv/-vvv: Increase verbosity

```
ansible-playbook -i inventory playbook.yml -K --check --diff -vv
```

GitHub: SupawitSaelim

Complete CLI Toolkit

- **ansible-config**

  View and manage Ansible configuration settings across environments

- **ansible-doc**

  Access comprehensive documentation for Ansible modules and plugins

- **ansible-galaxy**

  Manage Ansible roles and collections from the community

- **ansible-vault**

  Encrypt and decrypt sensitive data like passwords and API keys

- **ansible-pull**

  Pull playbooks from version control and run them locally on managed nodes

- **ansible-inventory**

  Display or dump configured inventory for debugging and verification

GitHub: SupawitSaelim

Host Patterns & Targeting

01

Multiple Hosts

Target multiple hosts using comma or colon: `web1.example.com,db1.example.com`

02

Exclusion Patterns

Use exclusion operator (!): `webservers:!web2.example.com`

03

Intersection Patterns

Use inclusion operator (&): `webservers:&dbservers`

04

Wildcard Matching

Match domains: `*.example.com` targets all hosts in domain

05

Array-like References

Use indices: `dbservers[0]`, `dbservers[-1]`, dbservers[0:2]

GitHub: SupawitSaelim

Configuration & Ad Hoc Commands

Configuration Management

Default config: `/etc/ansible/ansible.cfg`

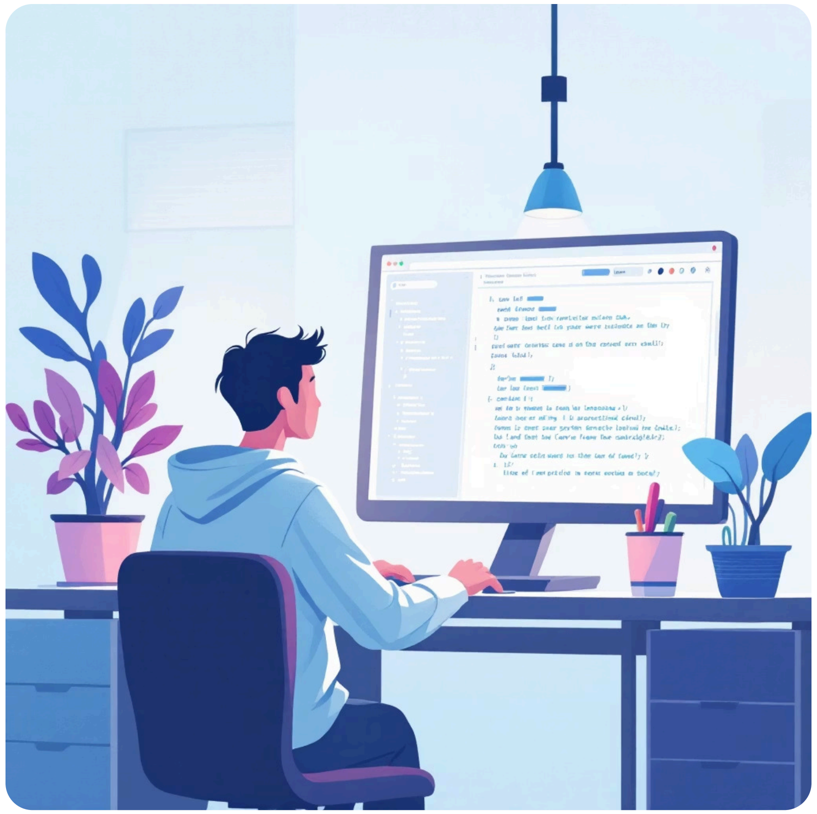
Generate complete template:

```
ansible-config init --disabled > ansible.cfg
```

This provides full configuration options with explanations for consistent team environments.

Practical Ad Hoc Examples

- **Reboot servers:** `ansible all -i inventory.ini -m reboot -u root -K`
- **Add users:** `ansible all -m user -a "name=ansible_user" -K`
- **Gather facts:** `ansible all -m setup -u root`



Ad hoc commands are perfect for one-off tasks like rebooting servers, managing users, or gathering system information quickly.