- ✓ DevOps & IaC
- ✓ Inventory & Ad-Hoc
- ✓ Playbooks & Modules
- ✓ Variables & Facts
- ✓ Handlers & Loops
- ✓ Jinja2 Templates
- ✓ Roles · Vault · Windows
- ✓ CI/CD · AWX

YAML

FULL COURSE

12 Lessons · Beginner to Advanced

Free · English & Dari

Learn it all @MyWebUniversity

The 12 Lessons



1. Welcome to Ansible and DevOps



2. Installing Ansible and Building a Lab



3. Inventory and Ad-Hoc Commands



4. Your First Playbook



5. Essential Modules



6. Variables, Facts and register



7. Conditionals, Loops and Handlers



8. Jinja2 Templates



9. Roles, Collections and Galaxy



10. Secrets, Privilege and Windows Hosts



11. Ansible in a DevOps Pipeline



12. Project: Deploy a Web App, and Next Steps



A FREE MYWEBUNIVERSITY COURSE

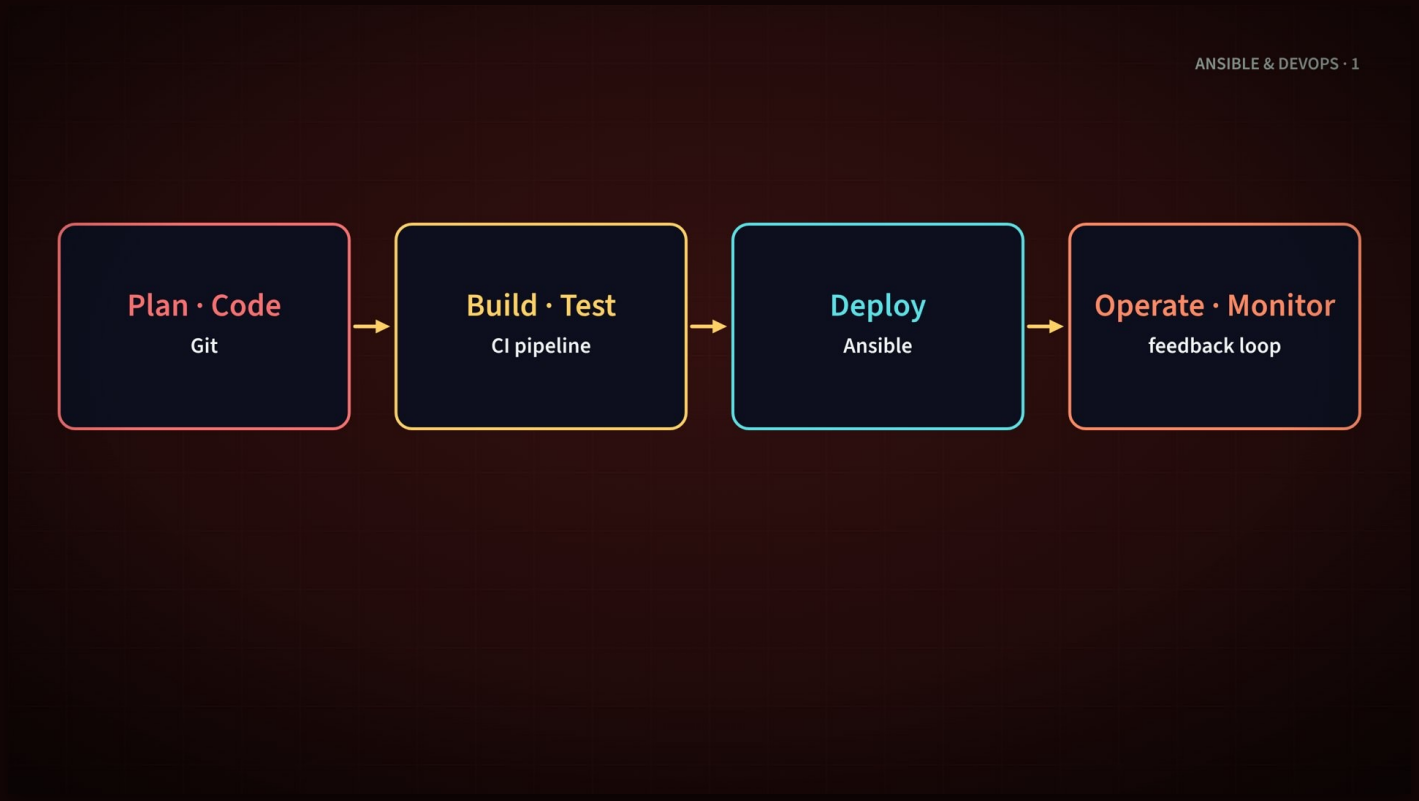
ANSIBLE & DEVOPS

LESSON 1

Welcome to Ansible and DevOps

What DevOps is, infrastructure as code, why Ansible, agentless push and idempotence

What Is DevOps?



- Plan · Code — Git
- Build · Test — CI pipeline
- Deploy — Ansible

1.01

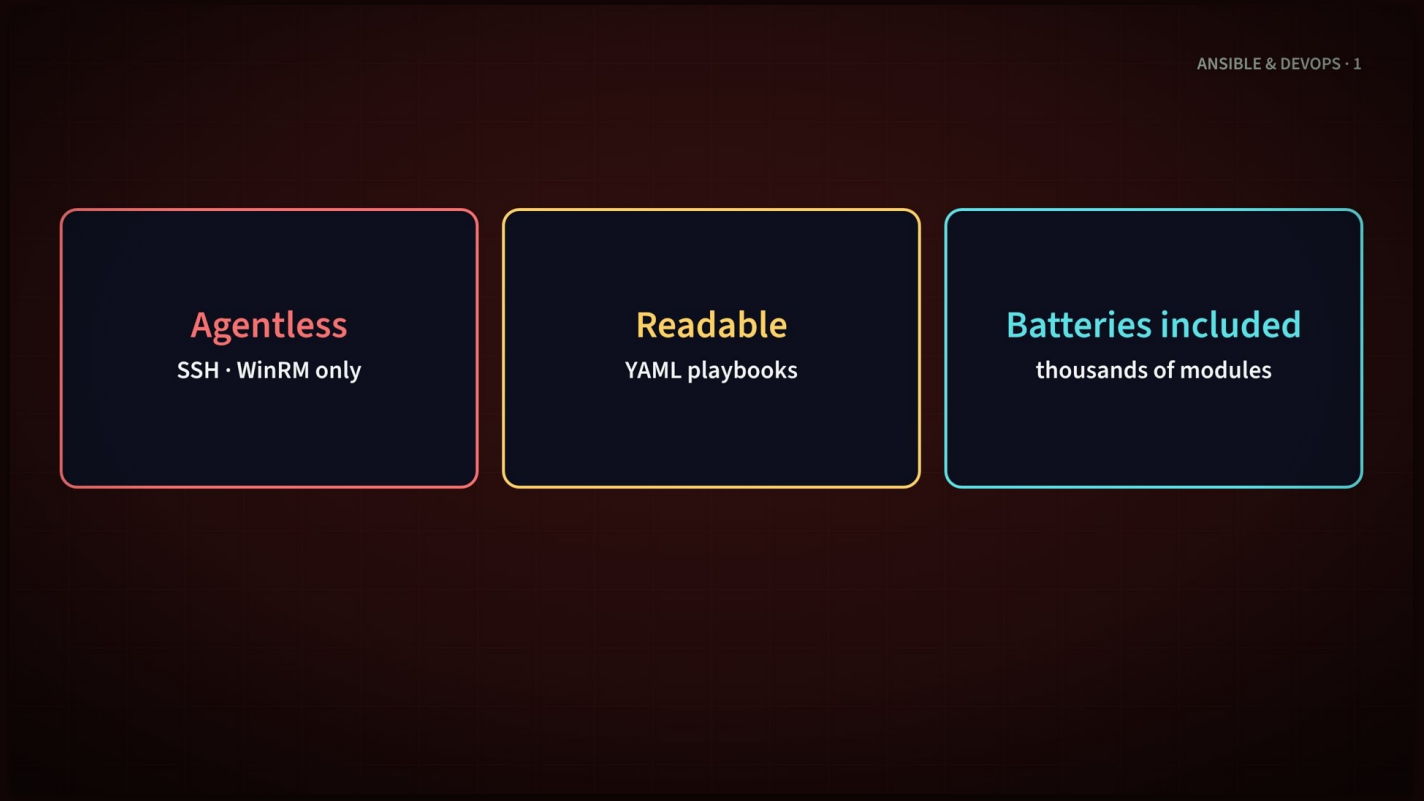
Infrastructure as Code



- Infrastructure as code means servers are configured by files kept in Git, not by clicking …
- Build ten identical servers, or rebuild one after a disaster, by running the same code.

1.02

Why Ansible?



- Agentless — SSH · WinRM only
- Readable — YAML playbooks
- Batteries included — thousands of modules

1.03

How Ansible Works



- Control node — ansible-playbook
- Inventory — web1, web2, db1
- SSH — push modules

1.04

Idempotence

terminal

```
$ ansible-playbook site.yml
PLAY RECAP
web1 : ok=4  changed=3  unreachable=0  failed=0
$ ansible-playbook site.yml
PLAY RECAP
web1 : ok=4  changed=0  unreachable=0  failed=0
```

- Ansible tasks describe the desired state, not the steps. Ensure nginx is installed; ensure …
- Run a playbook once and it makes changes. Run it again and it changes nothing, because …

1.05



A FREE MYWEBUNIVERSITY COURSE

ANSIBLE & DEVOPS

LESSON 2

Installing Ansible and Building a Lab

Control and managed nodes, installing with pip or dnf, SSH keys, ansible.cfg and the first

The Lab Design

Host	IP	Role
control	192.168.56.10	Ansible control node
web1	192.168.56.11	Rocky Linux, nginx
web2	192.168.56.12	Ubuntu, nginx
db1	192.168.56.13	Rocky Linux, PostgreSQL

- A lab needs one control node and a few managed nodes. Use virtual machines with VirtualBox …
- Our lab: control, two web servers and one database server, running Rocky Linux and Ubuntu.

2.01

Installing Ansible

terminal

```
$ python3 -m pip install --user ansible
$ sudo dnf install ansible-core        # RHEL, Rocky
$ sudo apt install ansible             # Ubuntu
$ ansible --version
ansible [core 2.17.5]
  python version = 3.12.3
```

- The control node must be Linux or macOS; on Windows, use WSL. Install with pip, or with the …
- ansible-core is the engine; the ansible package adds the community collections of modules.

2.02

SSH Keys

terminal

```
$ ssh-keygen -t ed25519 -C "ansible@control"
$ for h in web1 web2 db1; do ssh-copy-id ansible@$h; done
$ ssh ansible@web1 hostname
web1
```

- Ansible logs in with SSH, so set up key-based login. Create a key on the control node and …
- Use a dedicated user, such as ansible, with sudo rights on the managed nodes.

2.03

ansible.cfg

ansible.cfg

```
[defaults]
inventory         = ./inventory.ini
remote_user       = ansible
host_key_checking = True
forks             = 20

[privilege_escalation]
become        = True
become_method = sudo
```

- An ansible.cfg in your project folder sets defaults: where the inventory is, which user to …
- Ansible reads the file in the current folder first, so each project can have its own settings.

2.04

The First Command

terminal

```
$ ansible all -m ping
web1 | SUCCESS => { "ping": "pong" }
web2 | SUCCESS => { "ping": "pong" }
db1  | SUCCESS => { "ping": "pong" }
```

- The ping module tests that Ansible can log in and run Python on each host. It is not an …
- Green pong means success; red means check SSH, the user or Python on that host.

2.05



A FREE MYWEBUNIVERSITY COURSE

ANSIBLE & DEVOPS

LESSON 3

Inventory and Ad-Hoc Commands

INI and YAML inventories, groups, host and group variables, patterns and ad-hoc comma

An INI Inventory

inventory.ini

```
[web]
web1 ansible_host=192.168.56.11
web2 ansible_host=192.168.56.12

[db]
db1 ansible_host=192.168.56.13

[production:children]
web
db
```

- The inventory lists your hosts and puts them into groups in square brackets. A group of …
- Every host is automatically in the group named all.

3.01

A YAML Inventory

inventory.yml

```yaml
all:
  children:
    web:
      hosts:
        web1: { ansible_host: 192.168.56.11 }
        web2: { ansible_host: 192.168.56.12 }
    db:
      hosts:
        db1:  { ansible_host: 192.168.56.13 }
```

- The same inventory can be written in YAML, which is easier for large and nested structures.
- ansible-inventory with graph prints the tree, so you can check your groups.

3.02

host_vars and group_vars

terminal

```
$ tree
.
├── ansible.cfg
├── inventory.ini
├── group_vars/
│   ├── all.yml        # ntp_server: 192.168.56.1
│   └── web.yml        # http_port: 80
└── host_vars/
    └── web2.yml       # http_port: 8080
```

- Keep variables out of the inventory in folders named group_vars and host_vars, one YAML …
- Ansible loads them automatically, so all web servers can share an http_port while one host …

3.03

Host Patterns

Pattern	Targets
`all`	every host
`web`	the web group
`web:db`	web OR db
`production:&web`	in both groups
`all:!db1`	everything except db1
`web[0]`	first web host

- Patterns choose which hosts a command targets: a host, a group, all, or combinations. Colon …
- dash dash limit narrows any run further, which is a safe habit.

3.04

Ad-Hoc Commands

terminal

```
$ ansible web -m command -a "uptime"
web1 | CHANGED | rc=0 >> 10:42 up 3 days, load average: 0.05
$ ansible db -b -m package -a "name=htop state=present"
$ ansible all -m setup -a "filter=ansible_distribution*"
$ ansible web -b -m service -a "name=nginx state=restarted"
```

- An ad-hoc command runs one module on many hosts without writing a playbook. Use dash m for …
- Great for quick checks; anything you repeat belongs in a playbook.

3.05



A FREE MYWEBUNIVERSITY COURSE

ANSIBLE & DEVOPS

LESSON 4

Your First Playbook

YAML basics, plays and tasks, running playbooks, check and diff modes, reading the outp

YAML Basics

example.yml

```yaml
server:
  name: web1
  ports:
    - 80
    - 443
  enabled: true
  motd: "Welcome: authorized users only"
```

- YAML uses indentation with spaces, never tabs. A dash starts a list item, and key colon …
- Strings rarely need quotes, but quote anything that starts with curly braces or contains a …

4.01

Plays and Tasks

webserver.yml

```yaml
- name: Configure web servers
  hosts: web
  become: true
  tasks:
    - name: Install nginx
      ansible.builtin.package:
        name: nginx
        state: present

    - name: Start and enable nginx
      ansible.builtin.service:
        name: nginx
        state: started
        enabled: true
```

- A playbook is a list of plays. Each play targets hosts and holds a list of tasks. Each task …
- Use fully qualified module names like ansible.builtin.package, so it is always clear which …

4.02

Running a Playbook

terminal

```
$ ansible-playbook webserver.yml
TASK [Install nginx] ***************************
changed: [web1]
changed: [web2]
TASK [Start and enable nginx] ******************
changed: [web1]
changed: [web2]
PLAY RECAP *************************************
web1 : ok=3 changed=2 failed=0
```

- ansible-playbook runs it. Each task prints ok, changed, skipped or failed for every host. …
- If a task fails on one host, Ansible stops for that host and continues with the others.

4.03

Check, Diff and Syntax

terminal

```
$ ansible-playbook webserver.yml --syntax-check
$ ansible-playbook webserver.yml --list-hosts
$ ansible-playbook webserver.yml --check --diff
$ ansible-playbook webserver.yml --limit web1 -v
```

- Before touching production, test safely. syntax-check finds YAML mistakes. check mode is a …
- list-tasks and list-hosts preview what will run and where.

4.04

Multiple Plays

site.yml

```
- name: Common baseline
  ansible.builtin.import_playbook: common.yml

- name: Databases
  ansible.builtin.import_playbook: dbservers.yml

- name: Web tier
  ansible.builtin.import_playbook: webserver.yml
```

- One playbook file can hold several plays for different groups, run in order: first the …
- A site.yml that imports smaller playbooks is a common pattern for a whole environment.

4.05



A FREE MYWEBUNIVERSITY COURSE

ANSIBLE & DEVOPS

LESSON 5

Essential Modules

package, service, copy, file, user, lineinfile, command versus shell, and ansible-doc

Packages and Services

tasks.yml

```
- name: Install tools
  ansible.builtin.package:
    name: [git, curl, vim]
    state: present

- name: Firewall service running
  ansible.builtin.systemd:
    name: firewalld
    state: started
    enabled: true
```

- package works with any package manager; dnf and apt give more options. state present …
- service, or systemd, starts, stops, restarts and enables services.

5.01

copy and file

tasks.yml

```yaml
- name: Web root exists
  ansible.builtin.file:
    path: /var/www/school
    state: directory
    owner: nginx
    mode: "0755"

- name: Deploy home page
  ansible.builtin.copy:
    src: files/index.html
    dest: /var/www/school/index.html
    mode: "0644"
```

- copy sends a file from the control node, or writes inline content, and sets owner, group …
- Always quote octal modes, like 0644 in quotes, so YAML does not misread them.

5.02

Users and Lines in Files

tasks.yml

```
- name: Create deploy user
  ansible.builtin.user:
    name: deploy
    groups: wheel
    shell: /bin/bash

- name: Disable SSH root login
  ansible.builtin.lineinfile:
    path: /etc/ssh/sshd_config
    regexp: '^#?PermitRootLogin'
    line: 'PermitRootLogin no'
```

- user creates accounts with groups and shells; authorized_key installs their SSH keys. …
- These edit configuration safely without replacing whole files.

5.03

command vs shell

tasks.yml

```
- name: Initialize app database once
  ansible.builtin.command:
    cmd: /opt/app/bin/init-db
    creates: /opt/app/.db-initialized

- name: Count log errors
  ansible.builtin.shell: grep -c ERROR /var/log/app.log || true
  register: errors
  changed_when: false
```

- command runs a program directly; shell runs it through a shell, so pipes and redirection …
- Command tasks always report changed. Add creates, or changed_when, to keep them idempotent.

5.04

Finding Modules with ansible-doc

terminal

```
$ ansible-doc -l | grep firewall
ansible.posix.firewalld   Manage arbitrary ports/services with firewalld
community.general.ufw     Manage firewall with UFW
$ ansible-doc -s ansible.posix.firewalld
$ ansible-doc ansible.builtin.lineinfile
```

- ansible-doc shows the documentation and examples for every module, right in the terminal. …
- The examples section is usually the fastest way to learn a new module.

5.05



A FREE MYWEBUNIVERSITY COURSE

ANSIBLE & DEVOPS

LESSON 6

Variables, Facts and register

Defining variables, Jinja2 expressions, facts with setup, register, debug and precedenc

Defining Variables

app.yml

```
- hosts: web
  vars:
    app_name: school
    app_port: 8080
  tasks:
    - name: Create app directory
      ansible.builtin.file:
        path: "/opt/{{ app_name }}"
        state: directory

# ansible-playbook app.yml -e app_port=9090
```

- Variables can live in a play's vars section, in vars files, in group_vars and host_vars, or …
- Use them with double curly braces. When a value starts with a variable, quote the whole value.

6.01

Facts

terminal

```
$ ansible web1 -m setup -a "filter=ansible_distribution*"
"ansible_distribution": "Rocky",
"ansible_distribution_major_version": "9",
$ ansible web1 -m setup -a "filter=ansible_default_ipv4"
"address": "192.168.56.11"
```

- At the start of each play, Ansible gathers facts: operating system, IP addresses, memory, ...
- Use them to adapt, for example choosing the package name for Red Hat or Debian.

6.02

register and debug

check.yml

```yaml
- name: Check disk space
  ansible.builtin.command: df -h /
  register: disk
  changed_when: false

- name: Show result
  ansible.builtin.debug:
    msg: "{{ inventory_hostname }}: {{ disk.stdout_lines[1] }}"
```

- register saves a task's result into a variable: its output, return code and whether it …
- Together they let one task make decisions based on another.

6.03

Filters

filters.yml

```
- ansible.builtin.debug:
    msg:
      - "{{ app_env | default('production') }}"
      - "{{ app_name | upper }}"
      - "{{ groups['web'] | join(', ') }}"
      - "{{ ansible_memtotal_mb // 1024 }} GB RAM"
```

- Jinja2 filters transform values with a pipe: default gives a fallback, upper and lower …
- The default filter prevents errors when a variable is not defined.

6.04

Variable Precedence



- role defaults — weakest
- group_vars — all → groups
- host_vars · play vars — more specific

6.05



A FREE MYWEBUNIVERSITY COURSE

ANSIBLE & DEVOPS

LESSON 7

Conditionals, Loops and Handlers

when, loop, loop_control, handlers and notify, blocks with rescue, tags

Conditionals with when

os.yml

```yaml
- name: Install Apache on Red Hat family
  ansible.builtin.dnf:
    name: httpd
  when: ansible_facts['os_family'] == 'RedHat'

- name: Install Apache on Debian family
  ansible.builtin.apt:
    name: apache2
    update_cache: true
  when: ansible_facts['os_family'] == 'Debian'
```

- when runs a task only if a condition is true. Write the expression without curly braces. …
- The classic use: choose tasks by operating system family.

7.01

Loops

users.yml

```yaml
- name: Create team accounts
  ansible.builtin.user:
    name: "{{ item.name }}"
    groups: "{{ item.groups }}"
  loop:
    - { name: mariam, groups: wheel }
    - { name: ahmad,  groups: developers }
    - { name: zahra,  groups: developers }
  loop_control:
    label: "{{ item.name }}"
```

- loop repeats a task for every item in a list, and item holds the current value. Loop over …
- loop_control with label keeps the output short and readable.

7.02

Handlers

nginx.yml

```yaml
tasks:
  - name: Deploy nginx config
    ansible.builtin.template:
      src: nginx.conf.j2
      dest: /etc/nginx/nginx.conf
      validate: nginx -t -c %s
    notify: Restart nginx

handlers:
  - name: Restart nginx
    ansible.builtin.service:
      name: nginx
      state: restarted
```

- A handler is a task that runs only when notified, and only once, at the end of the play. ...
- If the configuration did not change, there is no restart, so the service stays undisturbed.

7.03

Blocks, rescue and always

deploy.yml

```yaml
- block:
    - name: Deploy new release
      ansible.builtin.unarchive:
        src: app-2.0.tar.gz
        dest: /opt/app
  rescue:
    - name: Restore previous release
      ansible.builtin.command: /opt/app/bin/rollback
  always:
    - name: Notify the team
      ansible.builtin.debug:
        msg: "Deploy finished on {{ inventory_hostname }}"
```

- block groups tasks so they share settings like become or when. rescue runs if any task in …
- Use it to roll back a failed deployment cleanly.

7.04

Tags

terminal

```
$ ansible-playbook site.yml --list-tags
      TASK TAGS: [config, packages, users]
$ ansible-playbook site.yml --tags config
$ ansible-playbook site.yml --skip-tags packages
```

- Tags label tasks so you can run only part of a playbook. dash dash tags runs the tagged …
- Tag tasks by purpose, such as packages, config and users.

7.05



A FREE MYWEBUNIVERSITY COURSE

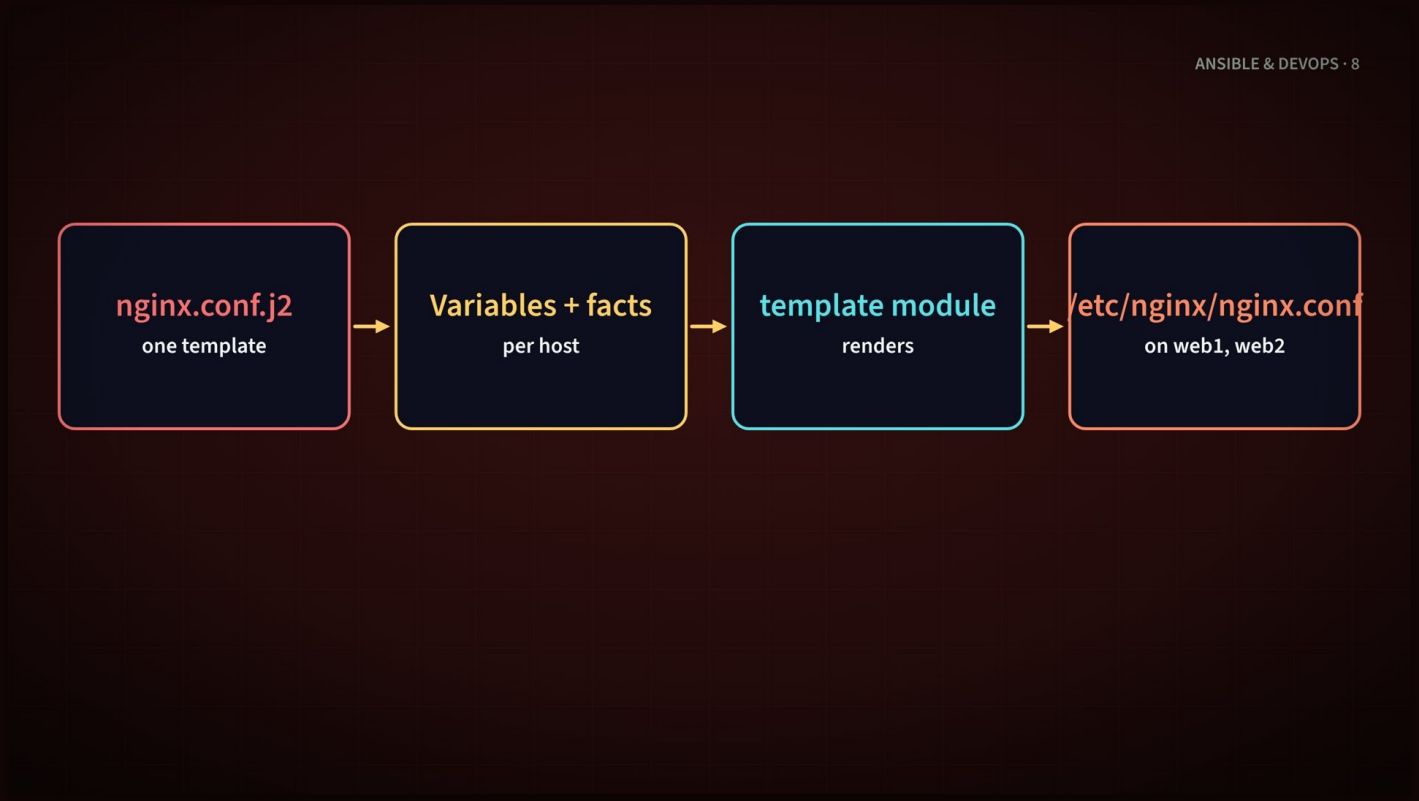
ANSIBLE & DEVOPS

LESSON 8

Jinja2 Templates

The template module, variables, loops and conditions in templates, whitespace and valida

Why Templates?



- nginx.conf.j2 — one template
- Variables + facts — per host
- template module — renders

8.01

Variables in Templates

templates/site.conf.j2

```
# {{ ansible_managed }}
server {
    listen {{ http_port }};
    server_name {{ inventory_hostname }}.school.edu;
    root /var/www/{{ app_name }};
    worker_connections {{ ansible_processor_vcpus * 1024 }};
}
```

- Inside a template, double curly braces print values, from variables or facts. Add a header …
- The template module copies the rendered result with owner and mode, and reports changed …

8.02

Loops and Conditions

templates/haproxy.cfg.j2

```
backend web_servers
    balance roundrobin
{% for host in groups['web'] %}
    server {{ host }} {{ hostvars[host]['ansible_default_ipv4']['address'] }}:{{ http_port }} check
{% endfor %}
{% if enable_stats | default(false) %}
listen stats
    bind *:8404
{% endif %}
```

- Statements use curly brace and percent: for loops repeat lines, and if blocks include lines …
- Here one template builds a load balancer list from every host in the web group.

8.03

The template Task

lb.yml

```yaml
- name: Configure HAProxy
  ansible.builtin.template:
    src: haproxy.cfg.j2
    dest: /etc/haproxy/haproxy.cfg
    owner: root
    mode: "0644"
    validate: haproxy -c -f %s
    backup: true
  notify: Reload haproxy
```

- The task names the source template and destination file. validate tests the new file before …
- backup yes keeps a timestamped copy of the previous file.

8.04

Whitespace and Testing

terminal

```
$ ansible-playbook lb.yml --check --diff
--- before: /etc/haproxy/haproxy.cfg
+++ after: haproxy.cfg.j2
@@ -4,3 +4,4 @@
     server web1 192.168.56.11:80 check
+    server web2 192.168.56.12:80 check
```

- Jinja2 statements can leave blank lines. A dash inside the tag, like curly percent dash, …
- Run with check and diff to see exactly how the rendered file will differ before applying it.

8.05



A FREE MYWEBUNIVERSITY COURSE

ANSIBLE & DEVOPS

LESSON 9

Roles, Collections and Galaxy

Role structure, defaults and vars, using roles, Ansible Galaxy, collections and requirements

Role Structure

terminal

```
$ ansible-galaxy role init roles/nginx
roles/nginx/
├── defaults/main.yml     # default variables
├── handlers/main.yml
├── tasks/main.yml
├── templates/
├── files/
├── vars/main.yml
└── meta/main.yml         # dependencies, platforms
```

- A role packages everything for one job, such as nginx or postgres, in a standard folder …
- ansible-galaxy role init creates the skeleton for you.

9.01

defaults vs vars

roles/nginx/defaults/main.yml

```yaml
# Port nginx listens on
nginx_port: 80
# Site name and document root
nginx_site_name: school
nginx_root: "/var/www/{{ nginx_site_name }}"
# Enable TLS (requires cert files)
nginx_tls: false
```

- defaults are meant to be overridden: ports, versions, feature switches. vars hold values …
- Document every default, because it is the role's public interface.

9.02

Using Roles

site.yml

```yaml
- hosts: web
  become: true
  roles:
    - common
    - role: nginx
      vars:
        nginx_port: 8080

- hosts: db
  become: true
  roles:
    - common
    - postgres
```

- List roles in a play's roles section, and pass variables to override defaults. Roles run …
- include_role and import_role call a role from inside the tasks list, even conditionally.

9.03

Ansible Galaxy and Collections

terminal

```
$ ansible-galaxy collection install community.general
$ ansible-galaxy collection list | grep -E 'posix|general'
ansible.posix        1.6.2
community.general    9.5.0
$ ansible-galaxy role install geerlingguy.docker
```

- Ansible Galaxy is the public hub for shared roles and collections. A collection bundles …
- Red Hat's Automation Hub offers certified, supported collections.

9.04

requirements.yml

requirements.yml

```
collections:
  - name: community.general
    version: ">=9.0.0"
  - name: ansible.posix
    version: 1.6.2
roles:
  - name: geerlingguy.docker
    version: 7.4.1

# ansible-galaxy install -r requirements.yml
```

- Pin every role and collection your project needs, with versions, in requirements.yml. Then …
- Review third-party roles before using them; they run with root privileges on your servers.

9.05



A FREE MYWEBUNIVERSITY COURSE

ANSIBLE & DEVOPS

LESSON 10

Secrets, Privilege and Windows Hosts

ansible-vault, no_log, become, managing Windows with WinRM and win_ modules

ansible-vault

terminal

```
$ ansible-vault create group_vars/db/vault.yml
$ ansible-vault edit group_vars/db/vault.yml
$ cat group_vars/db/vault.yml
$ANSIBLE_VAULT;1.1;AES256
623138633635626130383 6...
$ ansible-playbook site.yml --ask-vault-pass
```

- Never commit passwords in plain text. ansible-vault encrypts a whole file, or a single …
- Edit encrypted files with ansible-vault edit, and supply the password when running with …

10.01

A Clean Vault Pattern



- Keep two files per group: vars.yml with normal variables that point to vault_ variables, …
- Anyone can still search for which variables exist without decrypting anything. Add no_log …

10.02

become and Privilege

db.yml

```yaml
- name: Create application database
  community.postgresql.postgresql_db:
    name: school
  become: true
  become_user: postgres

- name: Create database user
  community.postgresql.postgresql_user:
    db: school
    name: "{{ db_user }}"
    password: "{{ db_password }}"
  become: true
  become_user: postgres
  no_log: true
```

- become runs tasks with elevated rights, by default with sudo as root. Set it per play or …
- Grant only what is needed: become at task level is safer than everywhere.

10.03

Managing Windows

group_vars/windows.yml

```
ansible_connection: winrm
ansible_port: 5986
ansible_winrm_transport: ntlm
ansible_winrm_server_cert_validation: validate
ansible_user: CORP\\ansible-svc
ansible_password: "{{ vault_win_password }}"

# ansible windows -m ansible.windows.win_ping
```

- Ansible manages Windows over WinRM, or SSH on newer systems. The control node still runs …
- win_ping tests the connection, just like ping for Linux.

10.04

Windows Modules

iis.yml

```
- hosts: windows
  tasks:
    - name: Install IIS
      ansible.windows.win_feature:
        name: Web-Server
        include_management_tools: true

    - name: Install security updates
      ansible.windows.win_updates:
        category_names: [SecurityUpdates]
        reboot: true
```

- The ansible.windows and community.windows collections provide win_feature for roles and …
- One playbook can patch and configure hundreds of Windows servers.

10.05



A FREE MYWEBUNIVERSITY COURSE

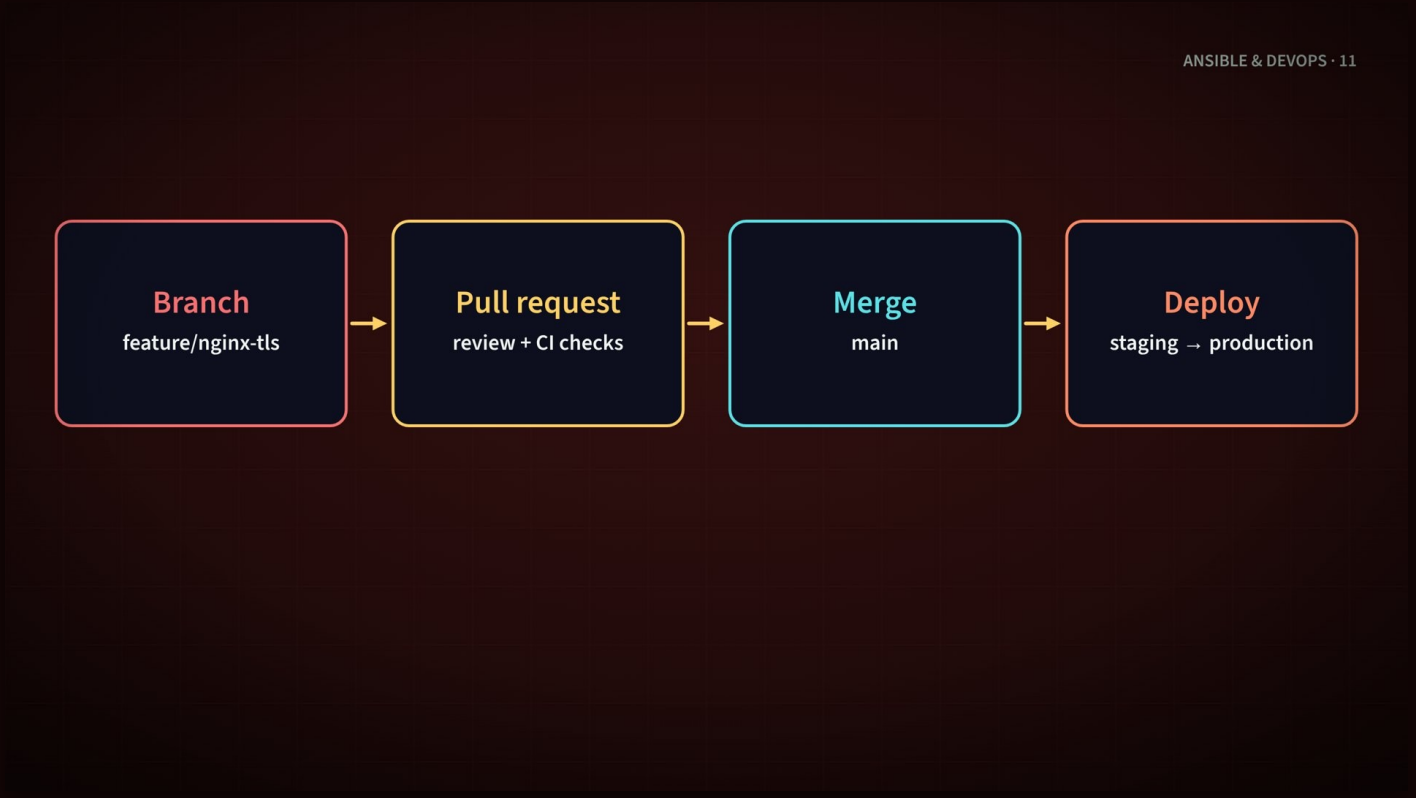
ANSIBLE & DEVOPS

LESSON 11

Ansible in a DevOps Pipeline

Git workflow, ansible-lint, Molecule testing, CI/CD with GitHub Actions or Jenkins, environ

Git Workflow



- Branch — feature/nginx-tls
- Pull request — review + CI checks
- Merge — main

11.01

ansible-lint

terminal

```
$ ansible-lint site.yml
name[missing]: All tasks should be named.
roles/nginx/tasks/main.yml:12 Task/Handler: shell grep ...
fqcn[action-core]: Use FQCN for builtin module actions (copy).
roles/nginx/tasks/main.yml:20
Failed: 2 failure(s), 0 warning(s)
```

- ansible-lint checks playbooks against best practices: missing names, short module names, …
- yamllint checks the YAML style. Run both on every commit.

11.02

Testing Roles with Molecule

terminal

```
$ molecule test
INFO  create      ✓  rocky9, ubuntu24
INFO  converge    ✓  changed=6
INFO  idempotence ✓  changed=0
INFO  verify      ✓  nginx answering on :80
INFO  destroy     ✓
```

- Molecule tests a role in throwaway containers or virtual machines. It creates the instance, …
- A role that passes Molecule on Rocky and Ubuntu is ready to share.

11.03

A CI/CD Pipeline

.github/workflows/deploy.yml

```yaml
on: { push: { branches: [main] } }
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - run: pip install ansible ansible-lint
      - run: ansible-lint
      - run: echo "$VAULT_PASS" > .vault_pass
        env: { VAULT_PASS: "${{ secrets.VAULT_PASS }}" }
      - run: ansible-playbook -i inventories/staging site.yml --vault-password-file .vault_pass
```

- A pipeline runs automatically on every push: lint, test, then deploy to staging, and to …
- Store the SSH key and vault password as pipeline secrets, never in the repository. Jenkins, …

11.04

Environments

terminal

```
inventories/
├── dev/
│   ├── hosts.ini
│   └── group_vars/all.yml      # app_replicas: 1
├── staging/
│   └── hosts.ini
└── production/
    ├── hosts.ini
    └── group_vars/all.yml      # app_replicas: 4
```

- Separate inventories for development, staging and production let the same playbooks run …
- Changes always flow in one direction: dev, then staging, then production. Never fix …

11.05



A FREE MYWEBUNIVERSITY COURSE

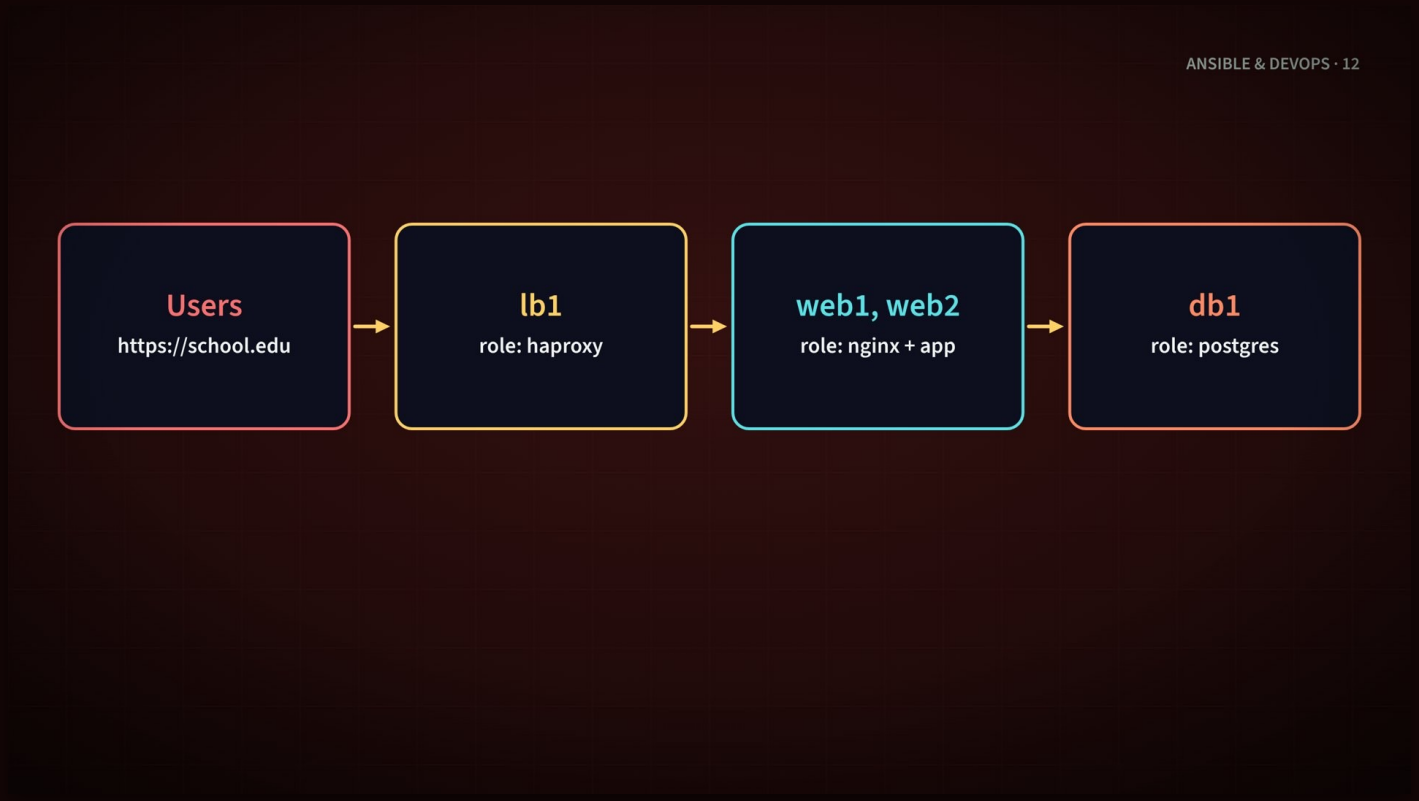
ANSIBLE & DEVOPS

LESSON 12

Project: Deploy a Web App, and Next Steps

A full multi-tier deployment, rolling updates, AWX and Automation Platform, and your pa

The Project



- Users — https://school.edu
- lb1 — role: haproxy
- web1, web2 — role: nginx + app

12.01

The Site Playbook

site.yml

```
- hosts: all
  become: true
  roles: [common, firewall]

- hosts: db
  become: true
  roles: [postgres]

- hosts: web
  become: true
  roles: [nginx, school_app]

- hosts: lb
  become: true
  roles: [haproxy]
```

- site.yml applies the common baseline everywhere, then the database, then the web tier, then …
- One command builds the whole environment from empty servers in minutes.

12.02

Zero-Downtime Rolling Updates

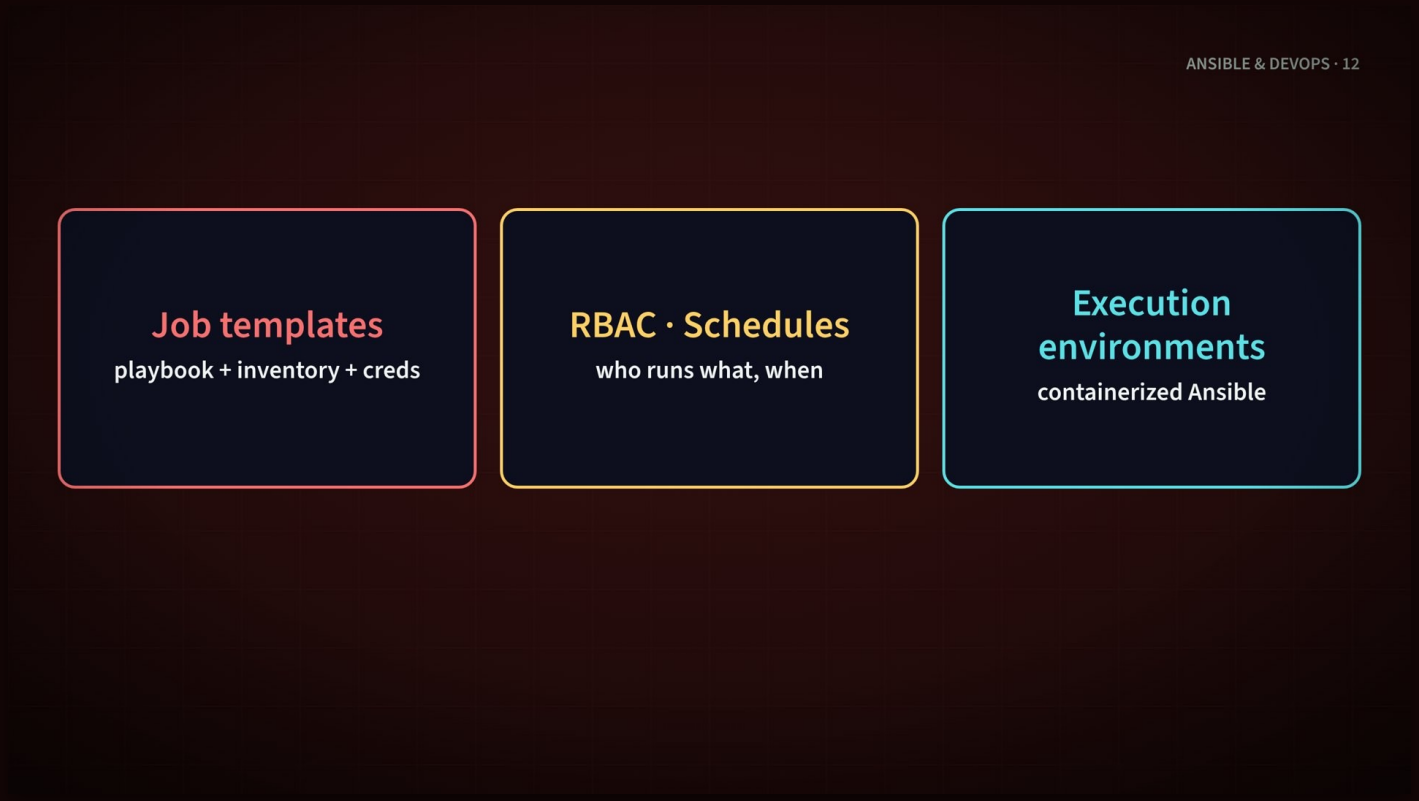
rolling.yml

```
- hosts: web
  serial: 1
  max_fail_percentage: 0
  pre_tasks:
    - name: Drain from load balancer
      community.general.haproxy:
        state: disabled
        host: "{{ inventory_hostname }}"
      delegate_to: lb1
  roles: [school_app]
  post_tasks:
    - name: Wait for health check
      ansible.builtin.uri: { url: "http://{{ inventory_hostname }}/health" }
    - name: Return to load balancer
      community.general.haproxy: { state: enabled, host: "{{ inventory_hostname }}" }
      delegate_to: lb1
```

- serial updates one host at a time. Before each host, delegate_to removes it from the load …
- max_fail_percentage stops the rollout if something goes wrong, so users never see an outage.

12.03

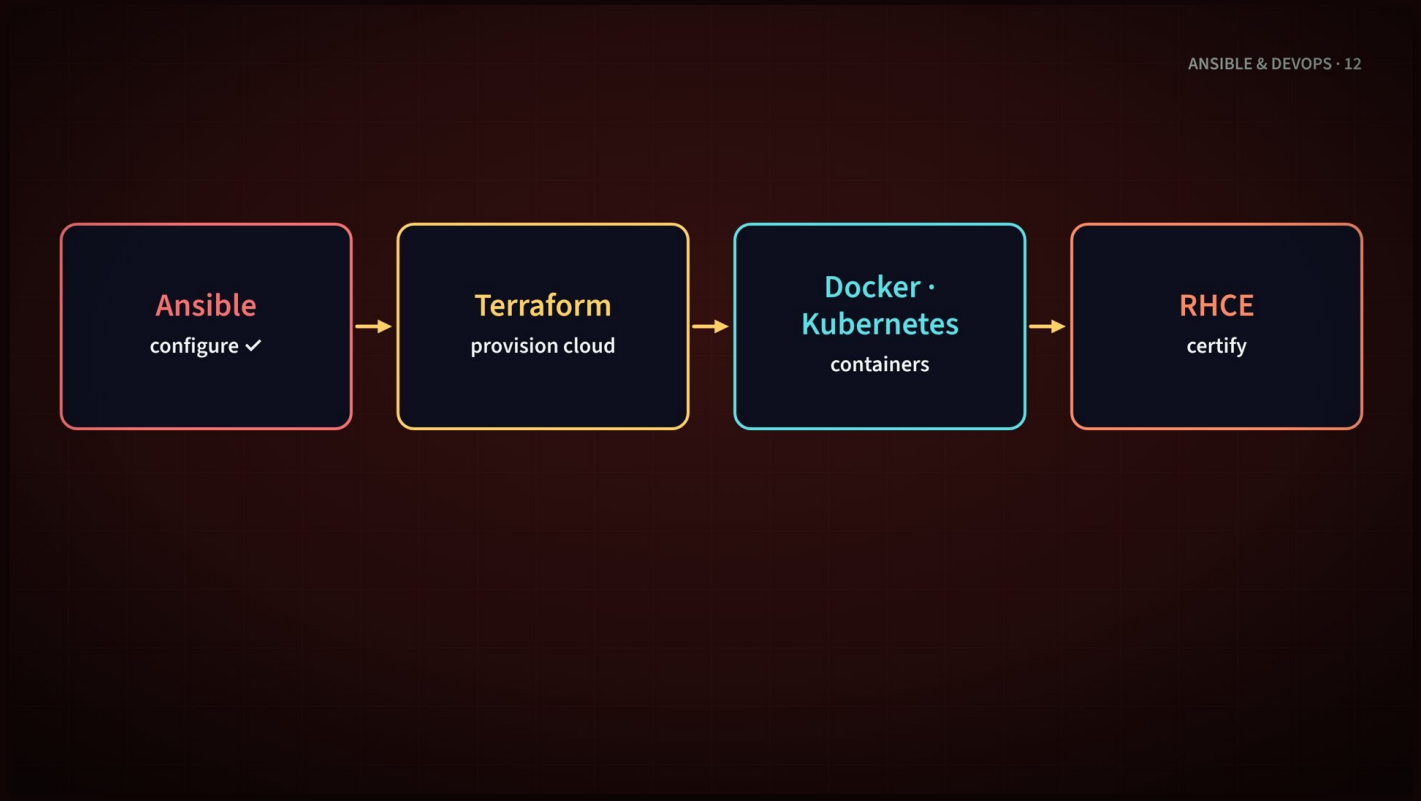
AWX and Ansible Automation Platform



- Job templates — playbook + inventory + creds
- RBAC · Schedules — who runs what, when
- Execution environments — containerized Ansible

12.04

Your DevOps Path



- Ansible — configure ✓
- Terraform — provision cloud
- Docker · Kubernetes — containers

12.05

Thank you for learning with us

MyWebUniversity.com

Learn it all @MyWebUniversity

Free technical education since 2000 — English and Dari