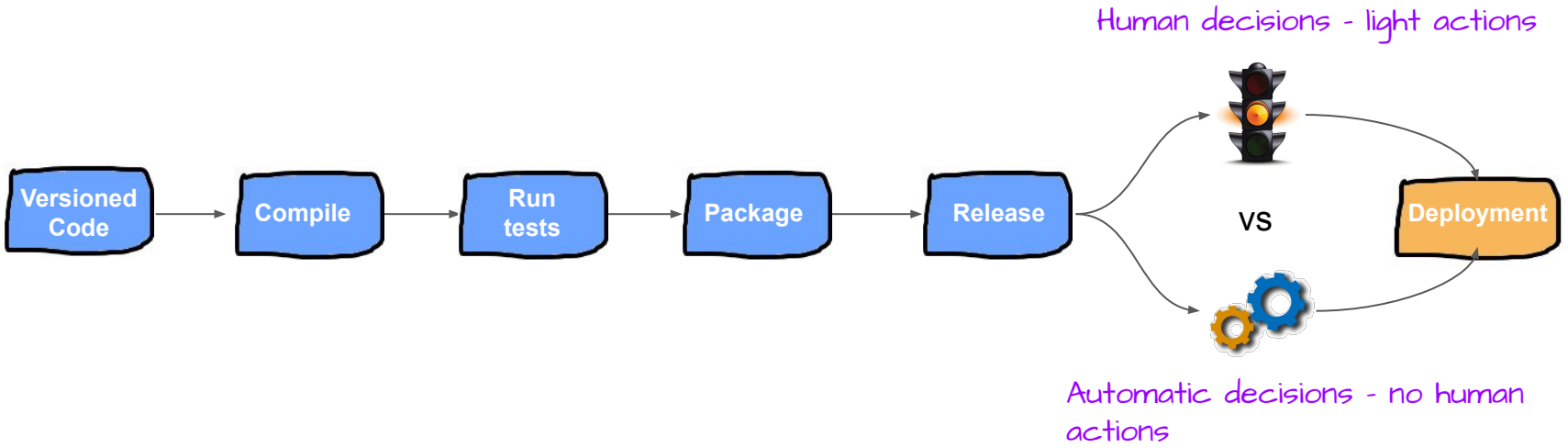
DevOps

Continuous Deployment & IaC with Ansible

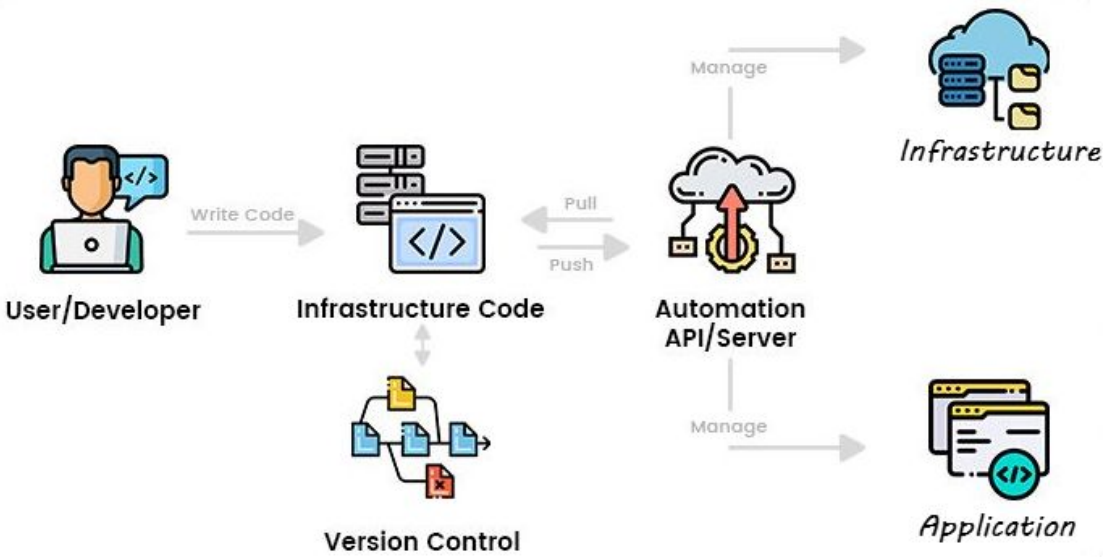
Reminders on Continuous Delivery



Reminders on Infrastructure As Code

- Consists of Managing and Provisioning infrastructure through machine-readable definition files.

- Key characteristics:
  - Automation
  - Version Control
  - Repeatable actions

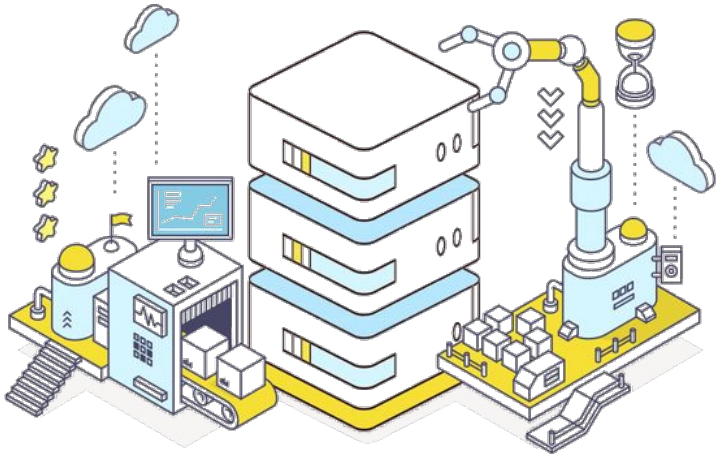


Ansible

- Open Source project, written in Python, sponsored by RedHat since 2015.
- Automation tool which can connect on remote hosts (ssh) and perform actions:
  - Install, deploy and configure tools
  - Run commands
  - …
- Agentless
- Repeatable actions (IaC) which can be run on multiple machines in parallel.
- Idempotent modules
- Uses YAML & Jinja2 to describe environment, actions and templates.

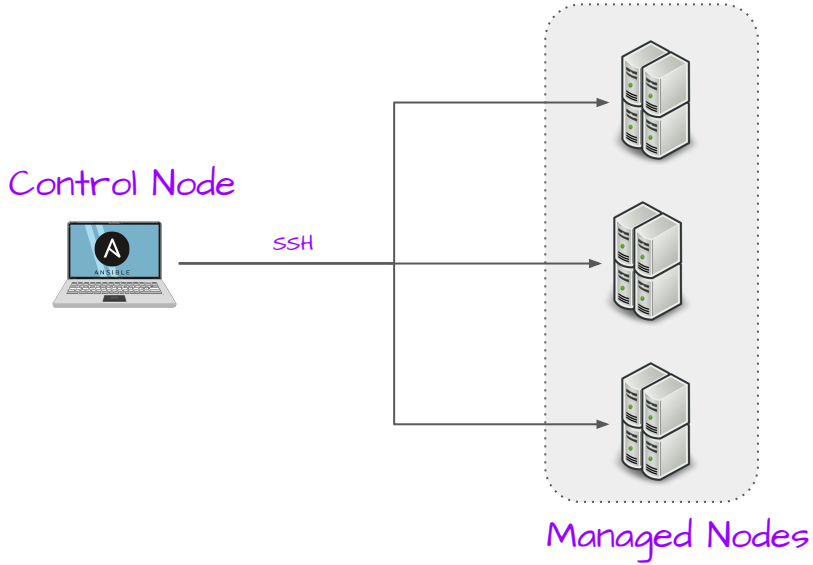
Ansible - Use cases

- Provisioning
  - Install & configure systems
  - Create Users & configure ACLs
- Patch management
  - Security updates
  - Version upgrades & migrations
- Application Deployment
  - Continuous delivery / deployment
- Tasks Orchestration
  - Conditions & tasks sequencing



Ansible Components

- **Inventories:** Describe your infra (hosts and groups)
- **Roles:** Sequential tasks to perform on hosts
- **Task:** Block defining a procedure to be executed
- **Variables:** Do not hardcode exact values which may change (ie: IP addresses, versions, ...). Centralize values which may change (ie: between environments)
- **Templates:** Files to deliver for which specific values depend on the delivery context.
- **Secrets:** Passwords, tokens, private keys, ...
- **Playbooks:** Entrypoint for Ansible. Describes which role should be played on which hosts.



Ansible Project Structure

```
inventory-prod.ini          # inventory file for production servers
inventory-preprod.ini       # inventory file for pre-production servers
inventory-dev.ini           # inventory file for development servers

group_vars/
    group1.yml          # here we assign variables to particular groups
    group2.yml

host_vars/
    hostname1.yml       # here we assign variables to particular systems
    hostname2.yml

roles/
    common/             # this hierarchy represents a "role"
        tasks/
            main.yml    # actions performed by the common role
    jdk/
    webapp/
    postgres/

site.yml                # playbook, runs site specific roles
webservers.yml          # playbook for webserver tier
dbservers.yml           # playbook for dbserver tier
```

Ansible Inventory

- List of managed nodes, optionally grouped.

- Ansible will use it to determine to which hosts roles should be applied.

- Hosts must be accessible through SSH (by default).

- Uses the ini syntax.

- Create 1 file per environment (dev / preprod / prod).

```
[proxy]
nginx-server        #may be a DNS name or an IP address

[master]
hadoop-namenode-1
hadoop-namenode-2

[worker]
hadoop-datanode-1
hadoop-datanode-2
hadoop-datanode-3
hadoop-datanode-4

[scheduler]
airflow-1

[jupyter]
edge-node-1
edge-node-2

[elastic]
elasticsearch-1
elasticsearch-2
elasticsearch-3
```

Ansible Roles

- Tasks may be divided in subgroups (roles) for modularity, maintenance and reusability.
- A role is a set of instructions which should be applied to specific host groups.
- Usually a role contains multiple subfolders:
  - vars : Role specific variables.
  - templates : Configuration files containing Jinja variables. The role is expected to fill and send them to hosts.
  - tasks : YAML describing instructions to be performed on hosts as steps. Variables can be interpreted in roles.
- Ansible comes with built-in modules used to describe tasks steps (ie: install a package, restart a service, delete files, ...). We can develop our own modules if needed.

Role to install and configure Nginx : 3 tasks + 1 template + variables

Role-specific variables

```
---
root_group: root
nginx_conf_path: {{ nginx_default_conf_path |
default('/etc/nginx/conf.d') }}
nginx_conf_file_path: /etc/nginx/nginx.conf
nginx_mime_file_path: /etc/nginx/mime.types
nginx_vhost_path: /etc/nginx/sites-enabled
```

Template

```
http {
{% block http_basic %}
  include       {{ nginx_mime_file_path }};
  default_type  application/octet-stream;
  client_max_body_size {{ nginx_client_max_body_size }};
  log_format  main  {{ nginx_log_format|indent(23) }};
  access_log  {{ nginx_access_log }};
  sendfile        {{ nginx_sendfile }};
  tcp_nopush      {{ nginx_tcp_nopush }};
  tcp_nodelay     {{ nginx_tcp_nodelay }};
  keepalive_timeout  {{ nginx_keepalive_timeout }};
  keepalive_requests {{ nginx_keepalive_requests }};
  server_tokens {{ nginx_server_tokens }};
{% if nginx_proxy_cache_path %}
  proxy_cache_path {{ nginx_proxy_cache_path }};
{% endif %}
{% endblock %}
}
```

Tasks

```
- name: Add PPA for Nginx (if configured).
  apt_repository:
    repo: 'ppa:nginx/{{ nginx_ppa_version }}'
    state: present
    update_cache: true
  when: nginx_ppa_use | bool

# Nginx setup.
- name: Copy nginx configuration in place.
  template:
    src: "{{ nginx_conf_template }}"
    dest: "{{ nginx_conf_file_path }}"
    owner: root
    group: "{{ root_group }}"
    mode: 0644
  notify:
    - reload nginx

- name: Ensure nginx service is running as configured.
  service:
    name: nginx
    state: "{{ nginx_service_state }}"
    enabled: "{{ nginx_service_enabled }}"
```

Module	Description	Example

apt	Package manager (Ubuntu / Debian)
	Install / Update / Remove packages
		`- name: Install latest version of "openjdk-8-jdk" ignoring "install-recommends"`
		`  apt:`
		`  name: openjdk-8-jdk`
		`  state: latest`
		`  install_recommends: no`

copy	Copy a file from local or remote machine to remote machine	`- name: Copy file from local to remote`
		`  copy:`
		`  src: /srv/myfiles/foo.conf`
		`  dest: /etc/foo.conf`
		`  remote_src: no`
		`  owner: foo`
		`  group: foo`
		`  mode: u+rw,g-wx,o-rwx`

unarchive	Untar a local or remote file to a remote machine	`- name: Unarchive a file that needs to be downloaded`
		`  unarchive:`
		`  src: https://example.com/example.zip`
		`  dest: /usr/local/bin`
		`  remote_src: yes`

service	Starts / Stops / Restart services	`- name: Restart service httpd`
		`  ansible.builtin.service:`
		`  name: httpd`
		`  state: restarted`

template	Deploy a local template file (jinja2) to remote after replacing variables	`- name: Template a file to /etc/file.conf`
		`  ansible.builtin.template:`
		`  src: /mytemplates/foo.j2`
		`  dest: /etc/file.conf`

git	Checkout code from Git	`- name: Git checkout`
		`  ansible.builtin.git:`
		`  repo: 'https://github.com/repo.git'`
		`  dest: /srv/checkout`
		`  version: release-0.22`

Ansible Playbooks

- This is where we describe what we want to do.

- Ordered list of roles to be executed on the specified hosts.

- May include tasks and variables (to avoid).

- Running an Ansible project consists of running playbooks.

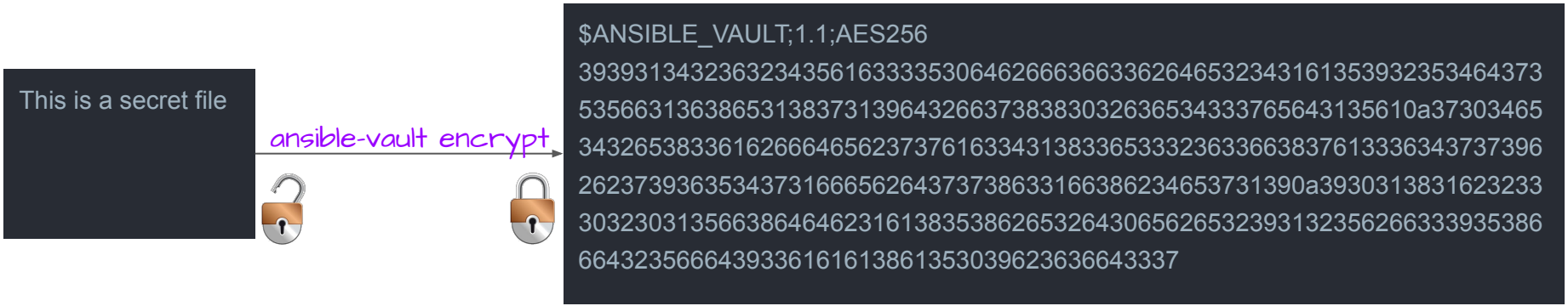
```yaml
---

- hosts: elastic
  remote_user: root
  become: yes
  become_method: sudo
  roles:
    - jdk
    - elasticsearch

- hosts: scheduler
  remote_user: root
  become: yes
  become_method: sudo
  roles:
    - python3
    - hadoop_conf
    - airflow
```

Ansible Vault

- Two ways to deal with secrets in Ansible projects:
  - Inject environment variables at runtime to the Ansible command.
  - Store them encrypted in the Ansible project using ansible-vault.



- Secrets can be safely versioned with your code (IAC).