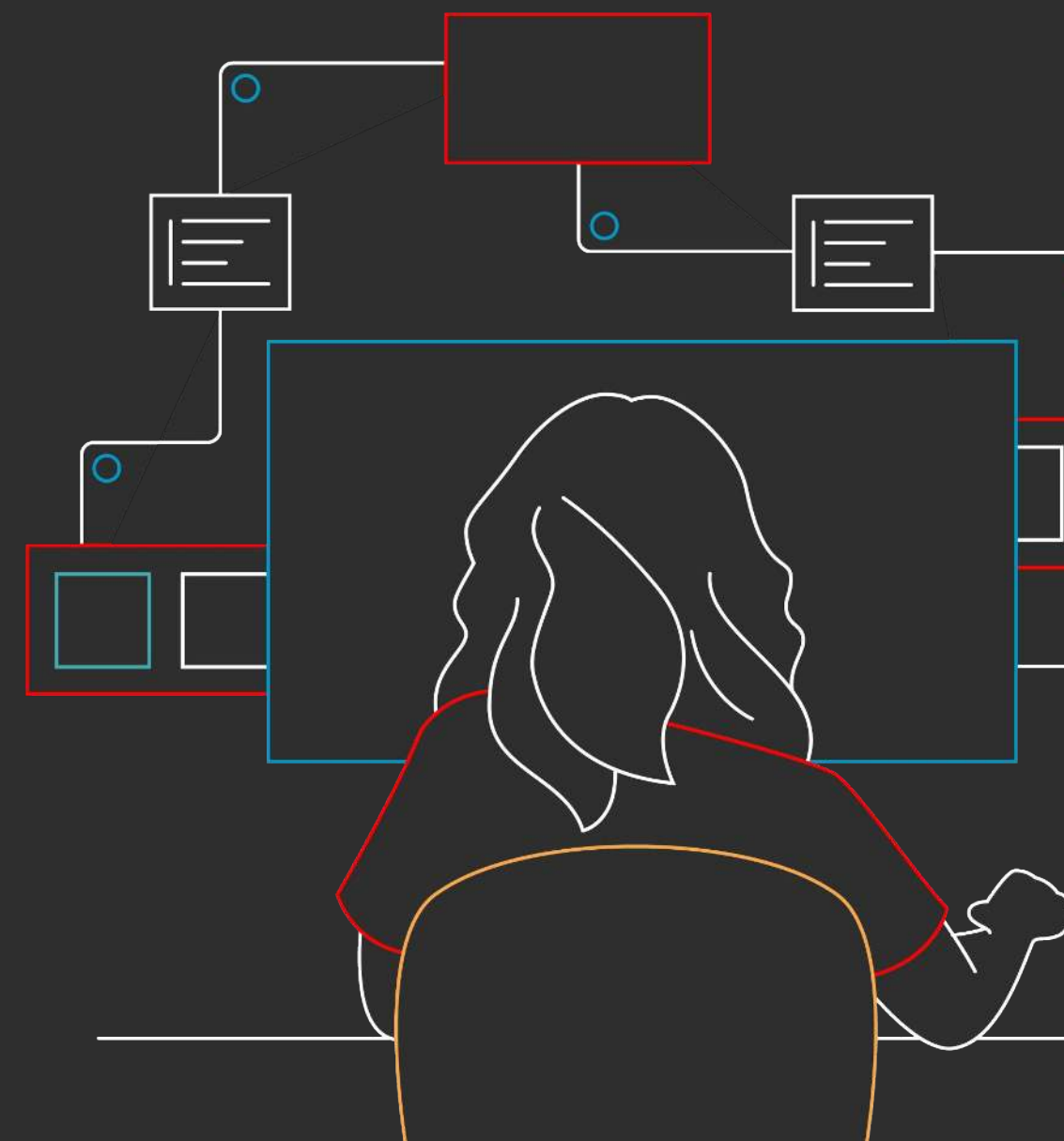
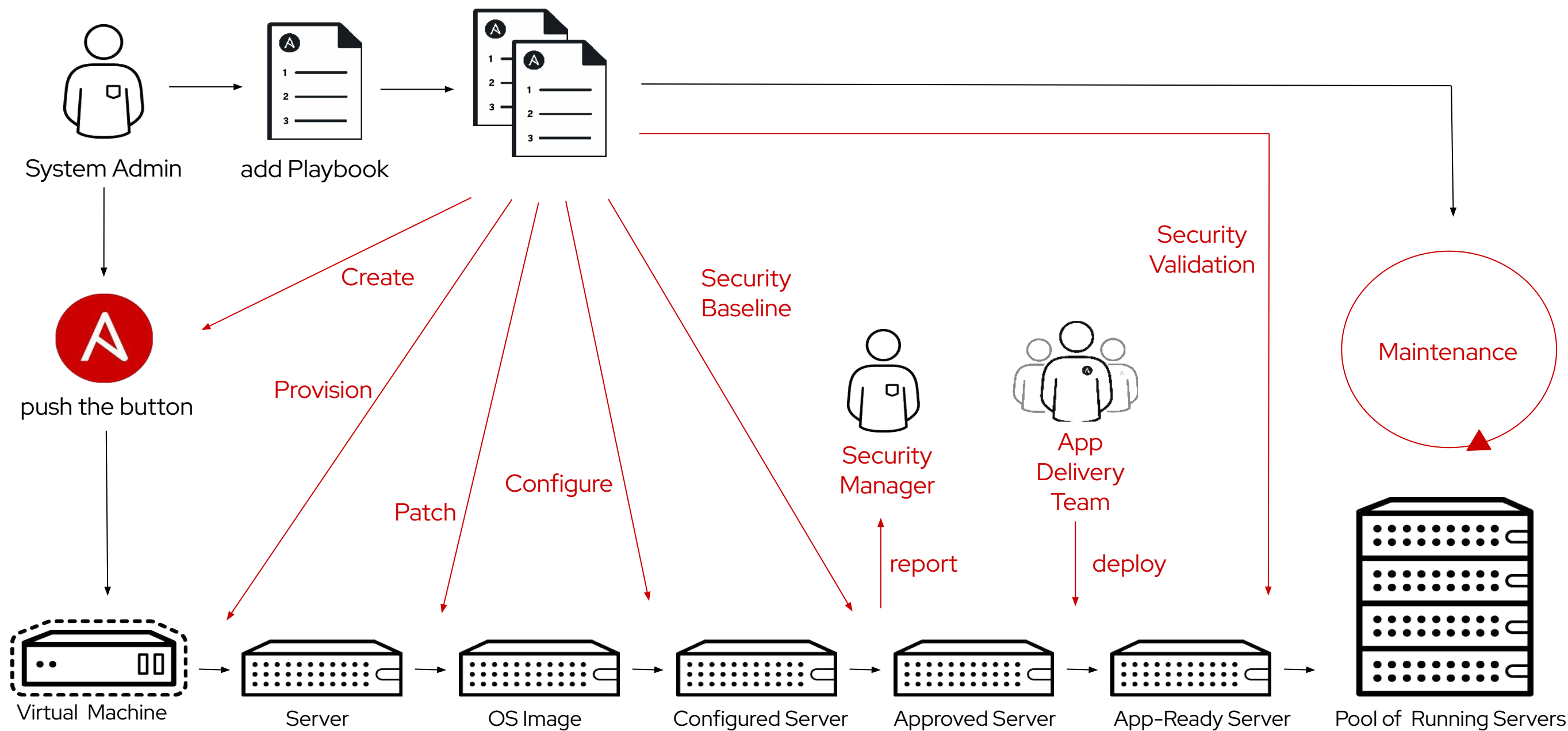


# Automating SAP Deployments with **Ansible**

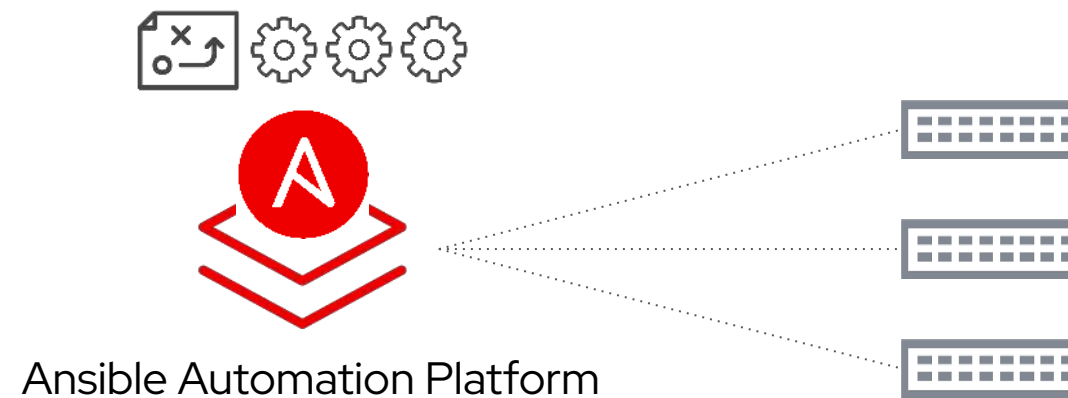
**Describing Ansible Concepts**



# Nothing routinary should be done manually



# Control Node and Execution Environment



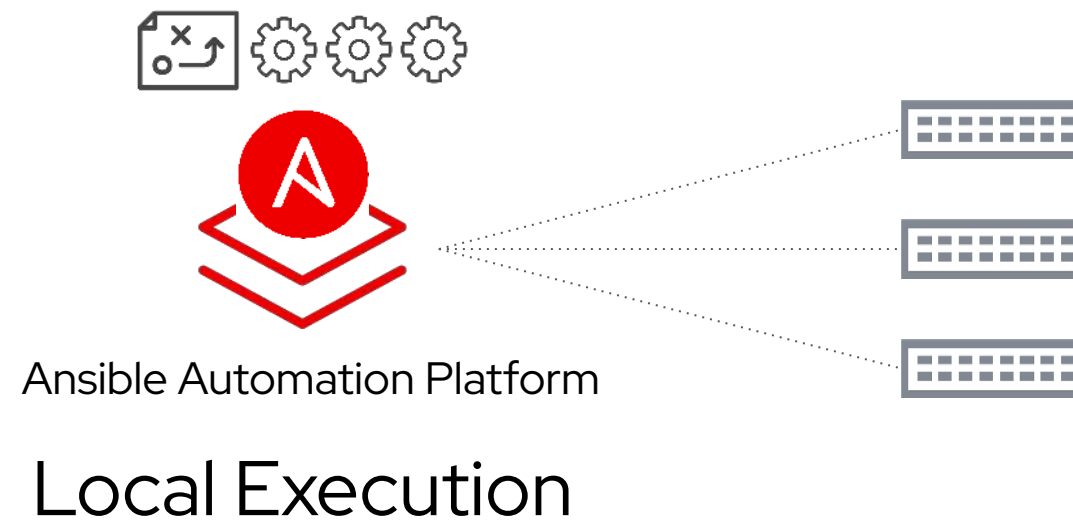
Control Node: host where automation content is stored

Execution Environment: container, that runs (executes) a playbook

configuration: `ansible.cfg` - ini style file for configuration of Ansible behaviour

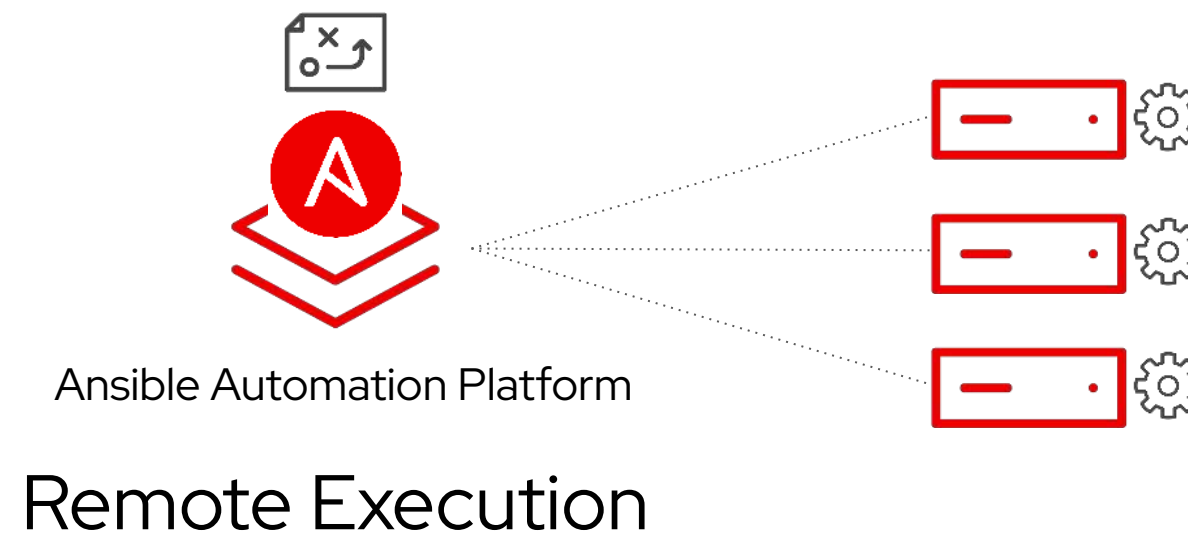
# How Ansible Automation Works

Module code is  
executed locally on the  
control node



Network Devices /  
API Endpoints

Module code is copied  
to the managed node,  
executed, then  
removed



Linux / Windows  
Hosts

# Ansible Inventory

The systems that a playbook runs against



## What are they?

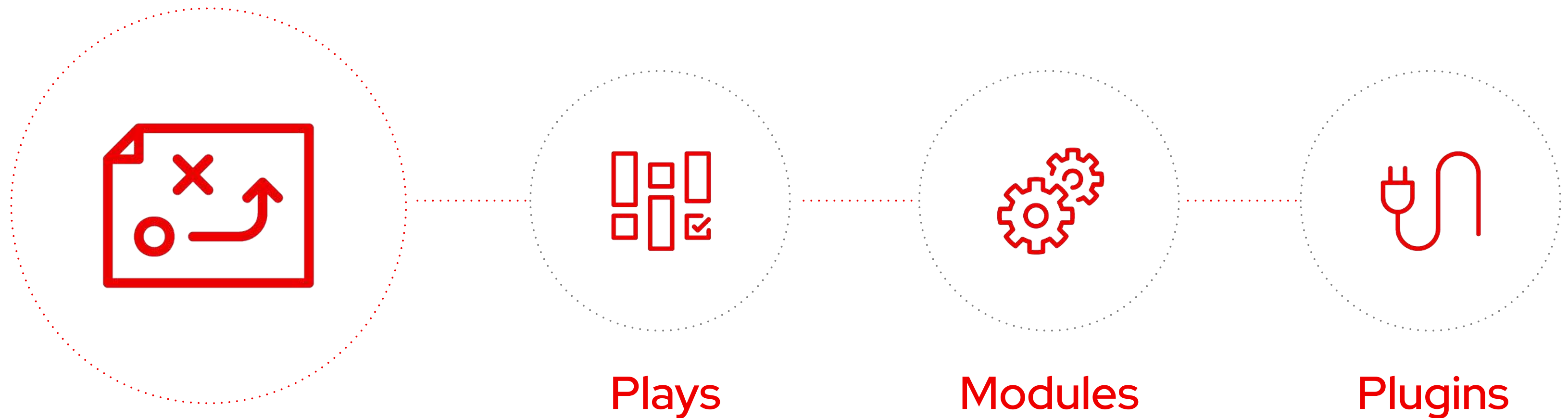
List of systems in your infrastructure that automation is executed against

```
[web]
webserver1.example.com
webserver2.example.com
```

```
[db]
dbserver1.example.com
```

```
[switches]
leaf01.internal.com
leaf02.internal.com
```

# What makes up an Ansible playbook?



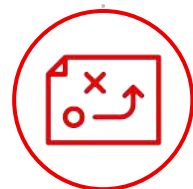
# Ansible plays

What am I automating?



## What are they?

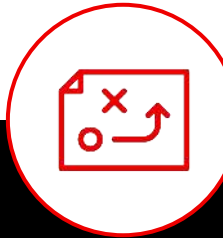
Top level specification for a group of tasks.  
Will tell that play which hosts it will execute on  
and control behavior such as fact gathering or  
privilege level.



## Building blocks for playbooks

Multiple plays can exist within an Ansible  
playbook that execute on different hosts.

```
---  
- name: install and start apache  
  hosts: web  
  become: yes
```



# A play

```
---
- name: install and start apache
  hosts: web
  become: yes

  tasks:
    - name: httpd package is present
      yum:
        name: httpd
        state: latest

    - name: latest index.html file is present
      template:
        src: files/index.html
        dest: /var/www/html/

    - name: httpd is started
      service:
        name: httpd
        state: started
```





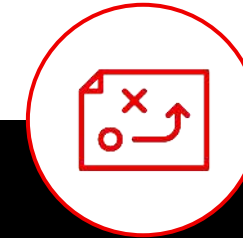
# A task

```
---
- name: install and start apache
  hosts: web
  become: yes

  tasks:
    - name: httpd package is present
      yum:
        name: httpd
        state: latest

    - name: latest index.html file is present
      template:
        src: files/index.html
        dest: /var/www/html/

    - name: httpd is started
      service:
        name: httpd
        state: started
```



# A module

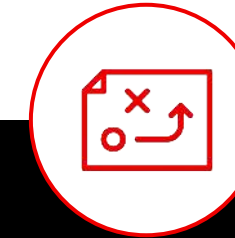


```
---
- name: install and start apache
  hosts: web
  become: yes

  tasks:
    - name: httpd package is present
      yum:
        name: httpd
        state: latest

    - name: latest index.html file is present
      template:
        src: files/index.html
        dest: /var/www/html/

    - name: httpd is started
      service:
        name: httpd
        state: started
```



# Running Playbooks

The most important **colors** of Ansible

**A task executed as expected, no change was made.**

**A task executed as expected, making a change**

**A task failed to execute successfully**

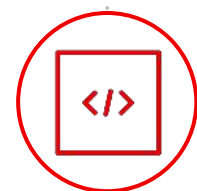
# Ansible modules

The “tools in the toolkit”



## What are they?

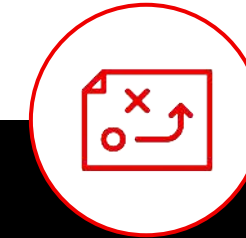
Parametrized components with internal logic, representing a single step to be done.  
The modules “do” things in Ansible.



## Language

Usually Python, or Powershell for Windows setups. But can be of any language.

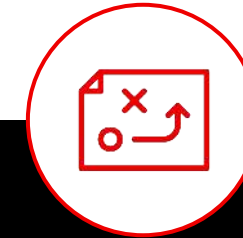
```
- name: latest index.html file ...  
  template:  
    src: files/index.html  
    dest: /var/www/html/
```



```
---
- name: variable playbook test
  hosts: localhost

  vars:
    var_one: awesome
    var_two: ansible is
    var_three: "{{ var_two }}" "{{ var_one }}"

  tasks:
    - name: print out var_three
      debug:
        msg: "{{ var_three }}"
```



```
---
- name: variable playbook test
  hosts: localhost

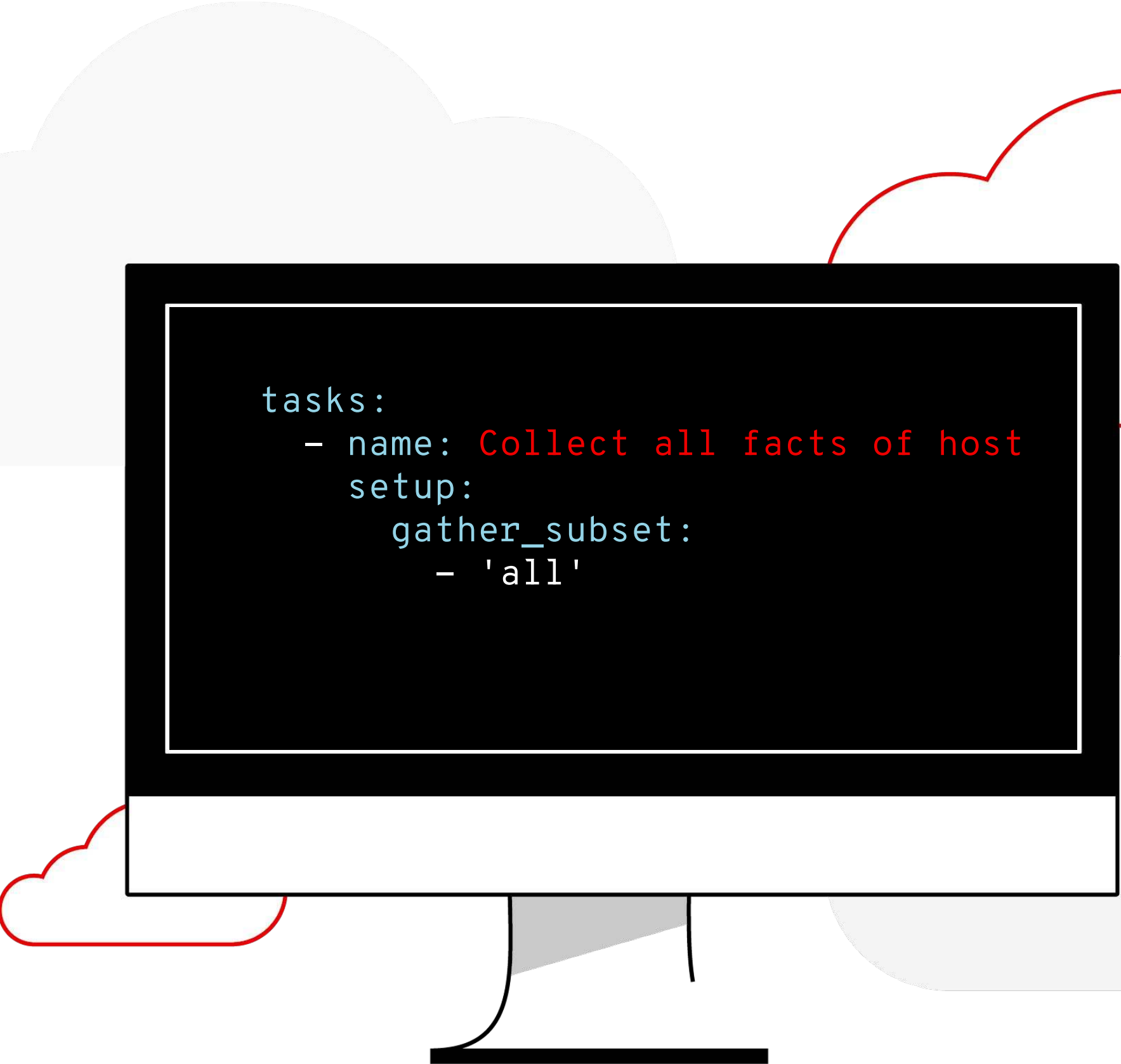
  vars:
    var_one: awesome
    var_two: ansible is
    var_three: "{{ var_two }} {{ var_one }}"

  tasks:
    - name: print out var_three
      debug:
        msg: "{{ var_three }}"
```

ansible is awesome

# Ansible Facts

- ▶ Just like variables, really...
- ▶ ... but: coming from the host itself!
- ▶ Check them out with the setup module



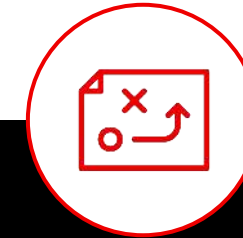
```
tasks:  
  - name: Collect all facts of host  
    setup:  
      gather_subset:  
        - 'all'
```

# Conditionals via VARS

Example of using a variable labeled *my\_mood* and using it as a conditional on a particular task.

```
vars:  
  my_mood: happy  
  
tasks:  
- name: task, based on my_mood var  
  debug:  
    msg: "Yay! I am {{ my_mood }}!"  
  when: my_mood == "happy"
```





```
---
- name: variable playbook test
  hosts: localhost

  vars:
    my_mood: happy

  tasks:
    - name: task, based on my_mood var
      debug:
        msg: "Yay! I am {{ my_mood }}"
      when: my_mood == "happy"
```

## Alternatively

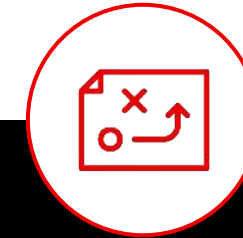
```
- name: task, based on my_mood var
  debug:
    msg: "Ask at your own risk. I'm {{ my_mood }}"
  when: my_mood == "grumpy"
```



```
---
- name: variable playbook test
  hosts: localhost

  tasks:
  - name: Install httpd
    yum:
      name: httpd
      state: latest
    when: ansible_distribution == 'RedHat'

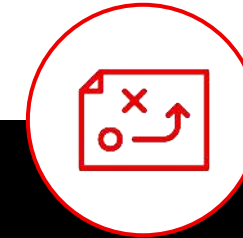
  - name: Install apache
    apt:
      name: apache2
      state: latest
    when: ansible_distribution == 'Debian' or
          ansible_distribution == 'Ubuntu'
```



```
---
- name: variable playbook test
  hosts: localhost

  tasks:
  - name: Ensure httpd package is present
    yum:
      name: httpd
      state: latest
      register: httpd_results

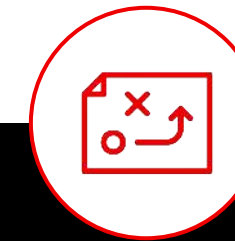
  - name: Restart httpd
    service:
      name: httpd
      state: restarted
    when: httpd_results.changed
```



```
---
- name: variable playbook test
  hosts: localhost

  tasks:
  - name: Ensure httpd package is present
    yum:
      name: httpd
      state: latest
    notify: restart_httpd

  handlers:
  - name: restart_httpd
    service:
      name: httpd
      state: restarted
```

**tasks:**

- name: Ensure httpd package is present
  - yum:
    - name: httpd
    - state: latest
    - notify: restart httpd
- name: Standardized index.html file
  - copy:
    - content: "This is my index.html file for {{ ansible\_host }}"
    - dest: /var/www/html/index.html
    - notify: restart httpd

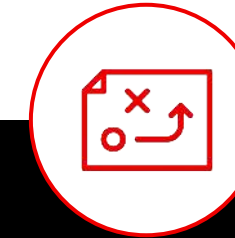
If **either** task notifies a **changed** result, the handler will be notified **ONCE**.

```
TASK [Ensure httpd package is present] *****
ok: [web2]
ok: [web1] unchanged
```

```
TASK [Standardized index.html file] *****
changed: [web2] changed
changed: [web1]
```

```
NOTIFIED: [restart_httpd] *****
changed: [web2]
changed: [web1]
```

**handler runs once**

**tasks:**

- name: Ensure httpd package is present  
yum:
  - name: httpd
  - state: latest
  - notify: restart httpd
- name: Standardized index.html file  
copy:
  - content: "This is my index.html file for {{ ansible\_host }}"
  - dest: /var/www/html/index.html
  - notify: restart httpd

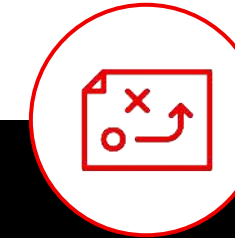
If **both** of these tasks notifies of a **changed** result, the handler will be notified **ONCE**.

```
TASK [Ensure httpd package is present] *****
changed: [web2]
changed: [web1] changed
```

```
TASK [Standardized index.html file] *****
changed: [web2]
changed: [web1] changed
```

```
NOTIFIED: [restart_httpd] *****
changed: [web2]
changed: [web1]
```

**handler runs once**

**tasks:**

- name: Ensure httpd package is present
  - yum:
    - name: httpd
    - state: latest
    - notify: restart httpd
- name: Standardized index.html file
  - copy:
    - content: "This is my index.html file for {{ ansible\_host }}"
    - dest: /var/www/html/index.html
    - notify: restart httpd

If **neither** task notifies a **changed** result, the handler ***does not run.***

TASK [Ensure httpd package is present] \*\*\*\*\*

ok: [web2]

ok: [web1]

**unchanged**

TASK [Standardized index.html file] \*\*\*\*\*

ok: [web2]

ok: [web1]

**unchanged**

PLAY RECAP \*\*\*\*\*

web2 : ok=2 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

web1 : ok=2 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0



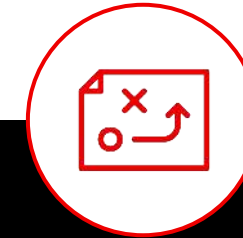
```
---
- name: Ensure users
  hosts: node1
  become: yes

  tasks:
    - name: Ensure user is present
      user:
        name: dev_user
        state: present

    - name: Ensure user is present
      user:
        name: qa_user
        state: present

    - name: Ensure user is present
      user:
        name: prod_user
        state: present
```





```
---
- name: Ensure users
  hosts: node1
  become: yes

  tasks:
    - name: Ensure user is present
      user:
        name: "{{item}}"
        state: present
      loop:
        - dev_user
        - qa_user
        - prod_user
```

# Variable Precedence

- role defaults (`roles/$ROLE/defaults/main.yml`)
- inventory vars (`vars/main.yml`)
- inventory group\_vars (`group_vars/$HOSTGROUP/*.yml`)
- inventory host\_vars (`host_vars/$FQDN/*.yml`)
- playbook group\_vars (we don't make a difference to inventory group\_vars)
- playbook host\_vars (we don't make a difference to inventory host\_vars)
- host facts (default facts of a host [Information discovered from system facts](#))
- play vars
- play vars\_prompt ([Prompts](#))
- play vars\_files (?)
- registered vars ([Register Variables](#))
- set\_facts ([Module set\\_fact](#))
- role and include vars (`roles/$ROLE/vars/main.yml`)
- block vars (only for tasks in block; [Blocks](#))
- task vars (only for the task)
- extra vars (always win precedence; `ansible --extra-vars='foo=bar'`)

# Ansible plugins

The “extra bits”



## What are they?

Plugins are pieces of code that augment Ansible’s core functionality. Ansible uses a plugin architecture to enable a rich, flexible, and expandable feature set.

### Example become plugin:

```
---  
- name: install and start apache  
  hosts: web  
  become: yes
```

### Example filter plugins:

```
{{ some_variable | to_nice_json }}  
{{ some_variable | to_nice_yaml }}
```

# Ansible roles

## Reusable automation actions



### What are they?

Group your tasks and variables of your automation in a reusable structure. Write roles once, and share them with others who have similar challenges in front of them.

```
---  
- name: install and start apache  
  hosts: web  
  roles:  
    - common  
    - webservers
```

# An Ansible Playbook

```
---
- name: manage chrony.conf
  hosts: hana
  become: yes

  vars:
    timeserver: time.example.org

  tasks:
    - name: Copy chrony configuration file
      template:
        src: chrony.conf.j2
        dest: /etc/chrony.conf
      notify:
        - restart_chronyd

  handlers:
    - name: restart_chronyd
      service:
        name: chronyd
        state: restarted
```

**update-chrony.yml**

# An Ansible Playbook

```
---
- name: manage chrony.conf
  hosts: hana
  become: yes

vars:
  timeserver: time.example.org
```

**update-chrony.yml**

```
tasks:
- name: Copy chrony configuration file
  template:
    src: chrony.conf.j2
    dest: /etc/chrony.conf
  notify:
    - restart_chronyd
```

```
handlers:
- name: restart_chronyd
  service:
    name: chronyd
    state: restarted
```

In the Filesystem:

```
$ ls
```

```
inventory
chrony-playbook.yml
chrony.conf.j2
```

## Creating Role: "update-chrony"

```
---  
- name: manage chrony.conf  
  hosts: hana  
  become: yes
```

**update-chrony.yml**

```
vars:  
  timeserver: time.example.org
```

```
$ mkdir roles; cd roles
```

```
$ ansible-galaxy init update-chrony
```

```
dest: /etc/chrony.conf  
notify:  
  - restart_chronyd
```

```
handlers:  
  - name: restart_chronyd  
    service:  
      name: chronyd  
      state: restarted
```

In the Filesystem:

```
$ ls
```

```
inventory
```

```
roles/update-chrony/tasks/main.yml  
                        /handlers/main.yml  
                        /templates/  
                        /files/  
                        /defaults/main.yml  
                        /vars/main.yml  
                        /meta/main.yml
```

# Move content from playbook to role

```
---  
- name: manage chrony.conf  
  hosts: hana  
  become: yes  
  
vars:  
  timeserver: time.example.org
```

**update-chrony.yml**

```
tasks:  
- name: Copy chrony configuration file  
  template:  
    src: chrony.conf.j2  
    dest: /etc/chrony.conf  
  notify:  
    - restart_chronyd
```

```
handlers:  
- name: restart_chronyd  
  service:  
    name: chronyd  
    state: restarted
```

## In the Filesystem:

```
$ ls
```

```
inventory  
chrony-playbook.yml  
chrony.conf.j2  
roles/
```

```
roles/update-chrony/  
    tasks/main.yml  
    /handlers/main.yml  
    /templates/chrony.conf.j2  
    /files/  
    /defaults/main.yml  
    /vars/main.yml  
    /meta/main.yml
```



# The playbook calling a role

```
---
- name: manage chrony.conf
  hosts: hana
  become: yes

  roles:
    - update-chrony
```

**update-chrony.yml**

```
---
timeserver: time.example.org
```

**main.yml**

```
---
- name: Copy chrony configuration file
  template:
    src: chrony.conf.j2
    dest: /etc/chrony.conf
  notify:
    - restart_chronyd
```

**main.yml**

```
---
- name: restart_chronyd
  service:
    name: chronyd
    state: restarted
```

**main.yml**

## In the Filesystem:

```
$ tree
├── chrony-playbook.yml
├── inventory
└── roles
    └── update-chrony
        ├── defaults
        │   └── main.yml
        ├── files
        ├── handlers
        │   └── main.yml
        ├── meta
        │   └── main.yml
        ├── tasks
        │   └── main.yml
        ├── templates
        │   └── chrony.conf.j2
        └── vars
            └── main.yml
```

## Additional Files

- Variables are stored in two locations
  - defaults/main.yml: least priority, i.e. default setting if not defined elsewhere
  - vars/main.yml: can only be overridden on the commandline with -e
- Meta Information of the role used by ansible galaxy:

```
galaxy_info:
  author: Markus Koch, Thomas Bludau
  description: Configures a RHEL OS to be ready for SAP HANA installation
  license: Apache 2.0
  min_ansible_version: 2.5
  platforms:
    - name: EL
      versions: [ 6, 7 ]

  galaxy_tags: [ 'system', 'sap', 'hana', 'beta' ]

dependencies: []
# List your role dependencies here, one per line. Be sure to remove the '[]' above,
# if you add dependencies to this list.
```

# Collections

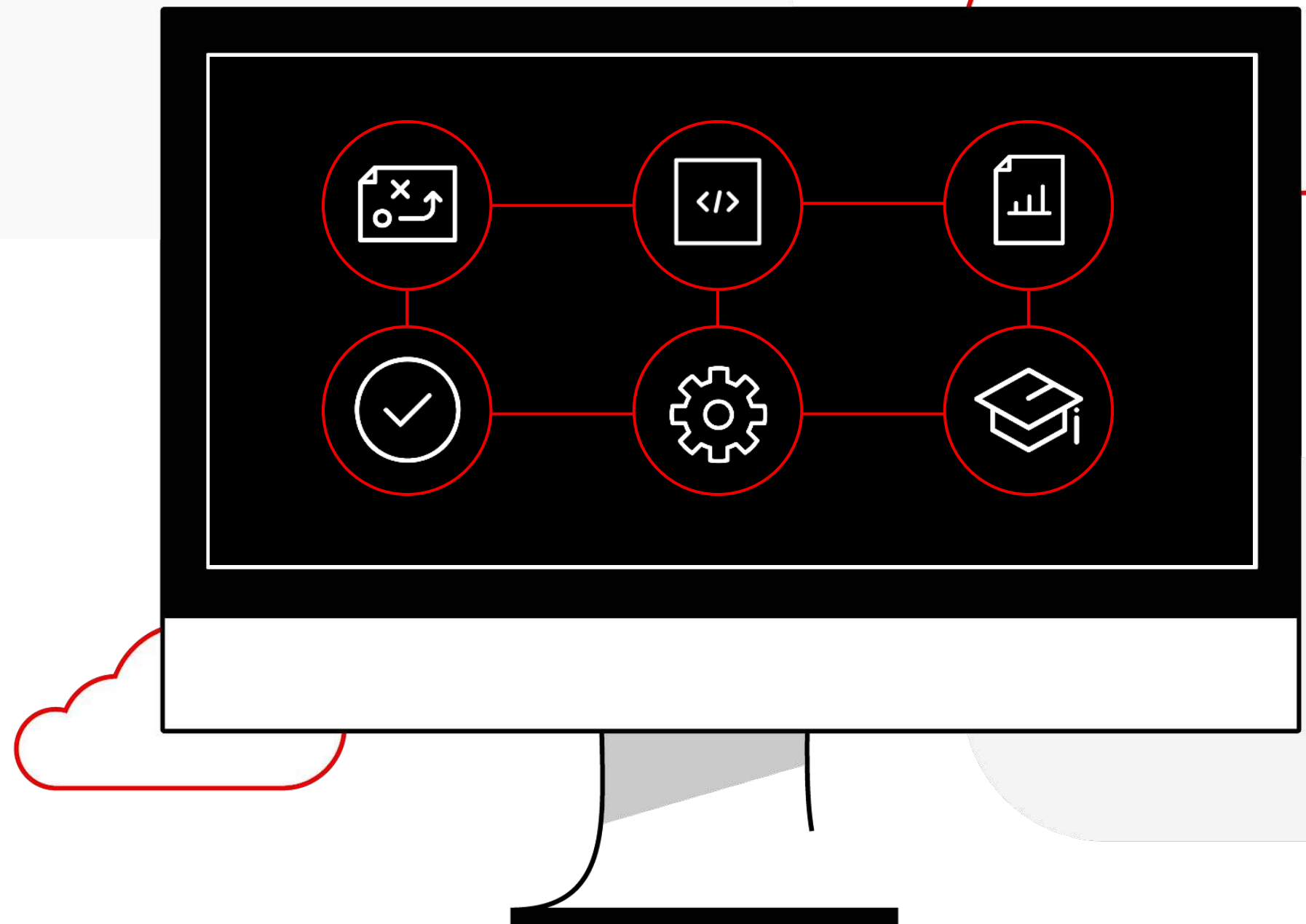
Simplified and consistent content delivery



## What are they?

Collections are a data structure containing automation content:

- ▶ Modules
- ▶ Playbooks
- ▶ Roles
- ▶ Plugins
- ▶ Docs
- ▶ Tests





nginx\_core

MANIFEST.json

playbooks

deploy-nginx.yml

...

plugins

README.md

roles

nginx

defaults

files

...

tasks

templates

...

nginx\_app\_protect

nginx\_config

deploy-nginx.yml

```
---
- name: Install NGINX Plus
  hosts: all
  tasks:
    - name: Install NGINX
      include_role:
        name: nginxinc.nginx
      vars:
        nginx_type: plus

    - name: Install NGINX App Protect
      include_role:
        name: nginxinc.nginx_app_protect
      vars:
        nginx_app_protect_setup_license: false
        nginx_app_protect_remove_license: false
        nginx_app_protect_install_signatures: false
```

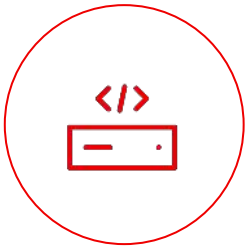
# Supported and certified content you can trust.

130+

Certified Content Collections

55+

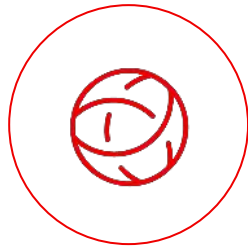
Certified technology partners



Infrastructure



Cloud



Network



Security

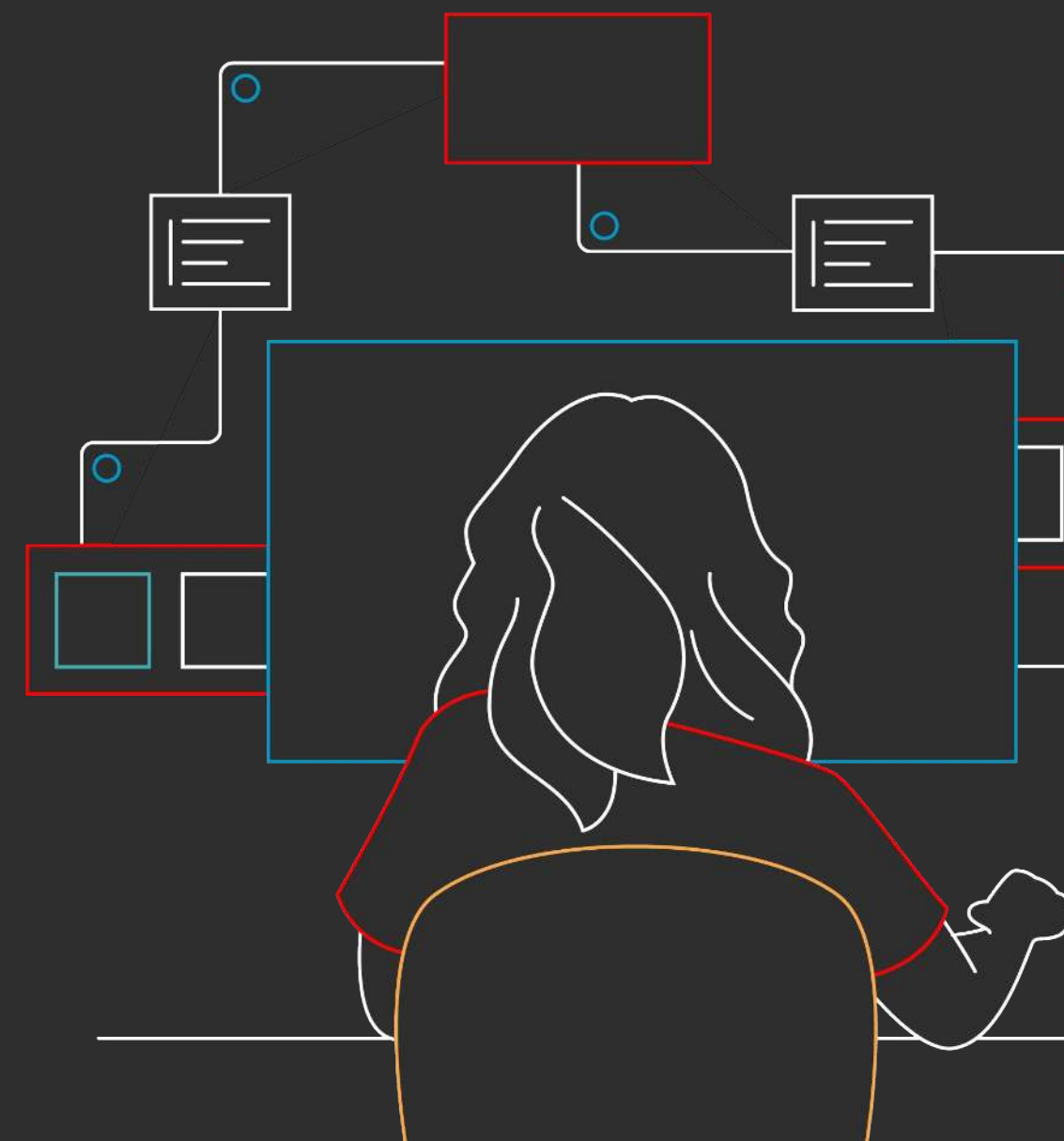


Edge

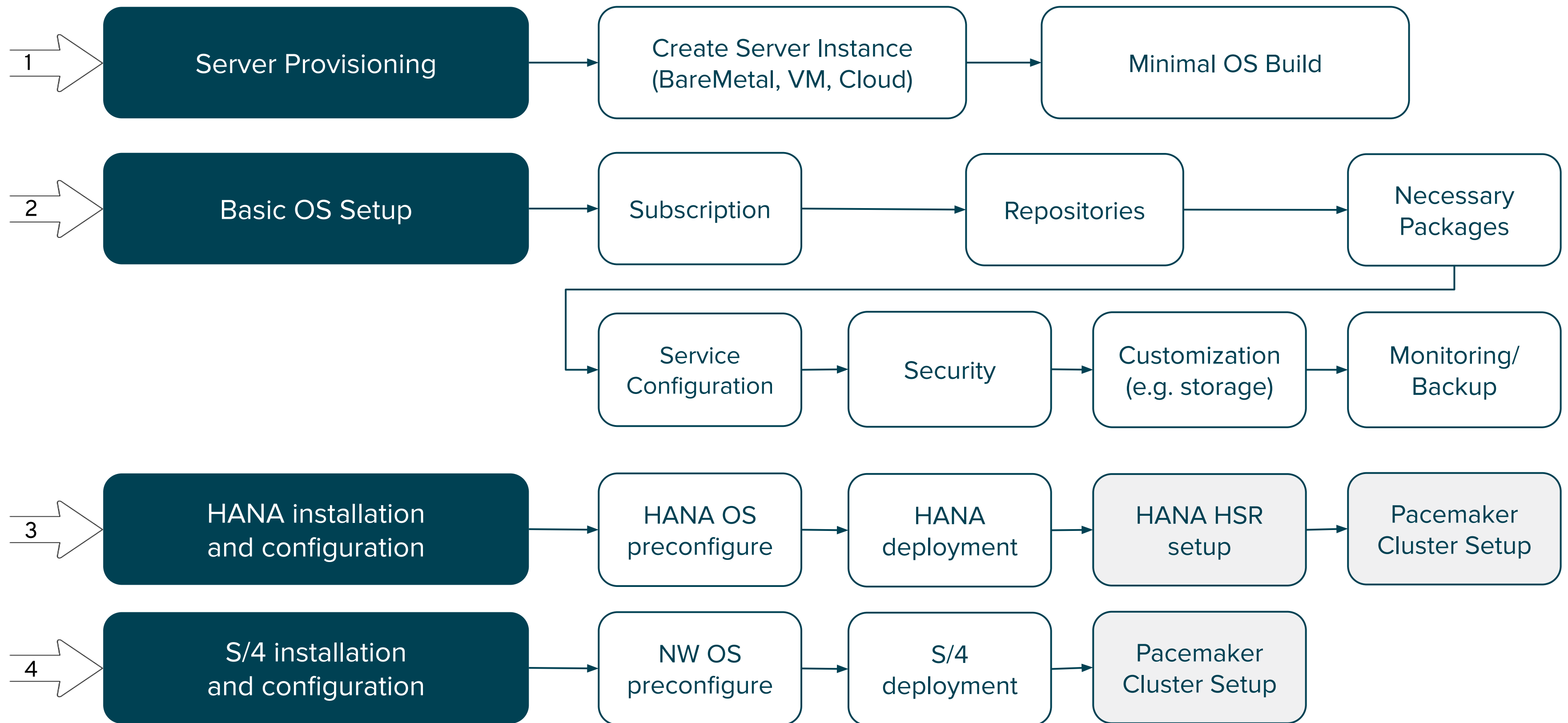


# Automating SAP Deployments with **Ansible**

**Installing and Configuring Ansible Roles for  
Base Configuration of SAP Systems**



# The SAP S/4 HANA deployment process breakdown



# Central Upstream repository

## SAP Linuxlab



A place for open source software that helps to make creating and managing SAP environments on Linux easier



<https://sap-linuxlab.github.io/>  
<https://github.com/sap-linuxlab>

- Terraform templates
- Ansible collections for
  - Software Download
  - Initial Install
  - Maintenance (Day 2 ops)
- Sizing Tools
- Reference Architectures



# Lab Time

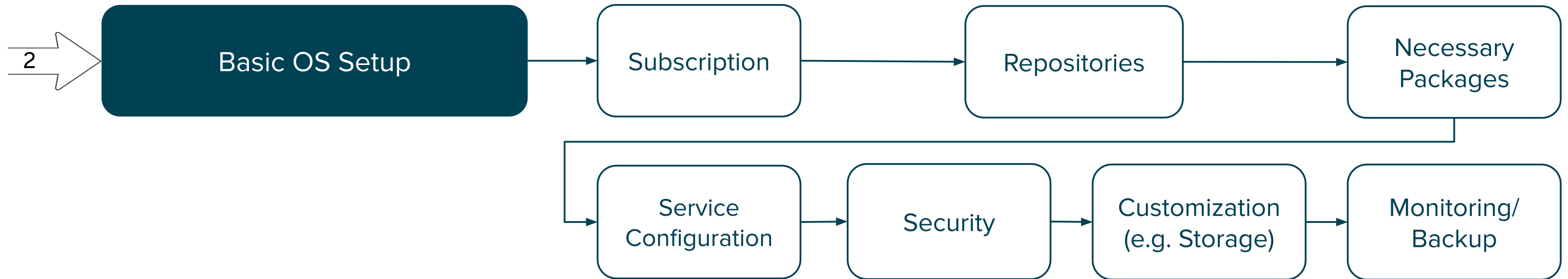
# The SAP S/4 HANA deployment process breakdown



- Provision can be solved in various ways
  - Bare Metal rack mounting and cabling + Satellite kickstart
  - Virtual environments with kickstart + image provisioning
  - public/private cloud with image provisioning
- multiple ways to reach this initial phase
  - ansible
  - terraform
  - custom scripts
  - manual process
- always ends up at least with a minimal RHEL system
  - with fully attached resources (Disk, CPU, memory)
  - reachable via ssh from ansible host

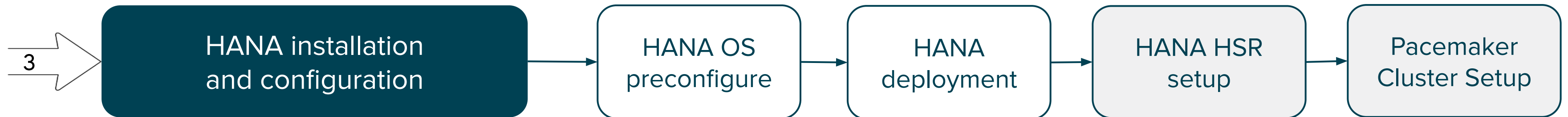
Phase 1 will not be covered  
in detail in this course

## The SAP S/4 HANA deployment process breakdown



- typically already in place “Corporate Standard Build”
- modules and roles exist in `rhel_system_roles` collection on Automation Hub (`linux_system_roles` on Galaxy)
- modules and roles to do these primary tasks, e.g.
  - `rhel_system_roles` / `linux_system_roles` (available also as RPM package)
    - [rhc](#)
    - [network](#)
    - [timesync](#)
    - [storage](#)
  - Ansible Galaxy or self-created collections for Monitoring and Backup

## The SAP S/4 HANA deployment process breakdown



- roles are available in [community.sap\\_install](#) collection
- HANA OS preconfigure
  - [sap\\_general\\_preconfigure](#)
  - [sap\\_hana\\_preconfigure](#)
- HANA deployment:
  - [sap\\_hana\\_install](#)
- HANA HSR setup
  - [sap\\_ha\\_install\\_hana\\_hsr](#)
- Pacemaker Cluster Setup
  - [sap\\_ha\\_pacemaker\\_cluster](#)

## The SAP S/4 HANA deployment process breakdown



- roles are available in [community.sap\\_install](#) collection
  - Netweaver OS preconfigure
    - [sap\\_general\\_preconfigure](#)
    - [sap\\_netweaver\\_preconfigure](#)
- S/4 deployment:
  - [sap\\_swpm](#)
- Pacemaker Cluster Setup (Optional)
  - [sap\\_ha\\_install\\_pacemaker](#) (TechPreview in 1.3.2)

# Role Details: Register system to satellite or RHN

**rhel\_system\_roles.rhc**

The first step is to make sure a system is registered and can access the correct repositories

This role is used to register against Satellite or RHN and configure the SAP repositories

## Example:

- Playbook:

```
- hosts: servers
  roles:
    - role: rhel_system_role.rhc
```

- Variables

```
rhc_auth:
  activation_keys:
    keys:
      - "{{ sap_rhsm_activationkey }}"
rhc_organization: "{{ sap_rhsm_org_id }}"
rhc_repositories: "{{ repositories }}"
rhc_release: "8.6"
rhc_state: "present"
rhc_insights:
  state: present
```

Configure Activation key [here](#)

## Role Details: Storage Setup

This role is very useful to configure complex disk setups

On the right side you see an example configuration of a HANA disk setup

```
storage_pools:
```

```
- name: sap
```

```
  disks:
```

```
    - xvdf
```

```
  volumes:
```

```
    - name: data
```

```
      size: "128 GiB"
```

```
      mount_point: "/hana/data"
```

```
      fs_type: xfs
```

```
      state: present
```

```
    - name: log
```

```
      size: "64 GiB"
```

```
      mount_point: "/hana/log"
```

```
      fs_type: xfs
```

```
      state: present
```

```
    - name: shared
```

```
      size: "256 GiB"
```

```
      mount_point: "/hana/shared"
```

```
      fs_type: xfs
```

```
      state: present
```

```
    - name: sap
```

```
      size: "50 GiB"
```

```
      mount_point: "/usr/sap"
```

```
      state: present
```

**rhel-system-roles.storage**

## Role Details: Storage Setup

**rhel-system-roles.storage**

This role is very useful to configure complex disk setups

On the right side you see an example configuration of a S/4HANA disk setup

```
storage_pools:
  - name: sap
    disks:
      - xvdf
    volumes:
      - name: sap
        size: "50 GiB"
        mount_point: "/usr/sap"
        state: present
      - name: sapmnt
        size: "20 GiB"
        mount_point: "/usr/sapmnt"
        state: present
      - name: swap
        size: "21 GiB"
        fs_type: swap
        mount_options: swap
        state: present
```



## Role Details: Configure Timeserver

**rhel-system-roles.timesync**

SAP requires proper time synchronisation.

So the linux system role is an easy way to set the time correctly

### Example:

- Playbook:

```
- hosts: servers
  roles:
    - role: rhel-system-roles.timesync
```

- Variables

```
timesync_ntp_servers:
  - hostname: 0.rhel.pool.ntp.org
    iburst: yes
timesync_ntp_provider: chrony
```

## Role Details: Networking Setup

**rhel-system-roles.network**

In most automatically deployed environments the network setup is done properly.

You could use `rhel-system-roles.network` to configure a more complex network preconfiguration. Simple configurations can also be done with `nmcli` module

### Example:

- Playbook:

```
- hosts: servers
  roles:
    - role: rhel-system-roles.network
```

- Variables

```
network_provider: nm
network_connections:
  - name: eth0
    #...
network_allow_restart: yes
```

## Role Details: Generic SAP settings

**redhat.sap\_install.  
sap\_general\_preconfigure**

SAP requires a couple of base settings that are described in [SAP Note 2369910](#) and other SAP notes which are required for all SAP systems. The role sap-preconfigure will set the parameters that have to be set for all SAP software.

The role is designed to be used without parameters to produce a valid output and has an assert mode, which can be used to verify the configuration

```
sap_general_preconfigure_modify_etc_hosts: true
sap_general_preconfigure_update: true
sap_general_preconfigure_fail_if_reboot_required: false
sap_general_preconfigure_reboot_ok: true
sap_hostname: myserver
sap_domain: example.com
sap_ip: 192.168.2.3
```

## Role Details: configure SAP Netweaver Settings

**redhat.sap\_install.  
sap\_netweaver\_preconfigure**

This role does all preconfiguration steps for SAP Netweaver which are described in SAP Note 2772999 for RHEL8.

It can be used without any additional parameters. It automatically fails, if not enough swap space is configured.

If you require different swap space than the recommended, you have variables to influence this setting.

## Role Details: configure SAP HANA Settings

**redhat.sap\_install.  
sap\_hana\_preconfigure**

This role performs the configurations according to the necessary SAP Notes

This role can be used without any additional parameters, although there are some that might be tweaked in production. e.g. some kernel parameters.

[SAP NOTE 2382421](#) defines a lot of kernel parameter options, that can be set, in the variable `sap_hana_preconfigure_kernel_parameters`.

```
sap_hana_preconfigure_set_minor_release: true
sap_hana_preconfigure_update: true
sap_hana_preconfigure_reboot_ok: true
sap_hana_preconfigure_fail_if_reboot_required: false
```



# Thank You !



<https://linkedin.com/company/Red-Hat>



<https://facebook.com/RedHatinc>



<https://youtube.com/user/RedHatVideos>



<https://twitter.com/RedHat>

