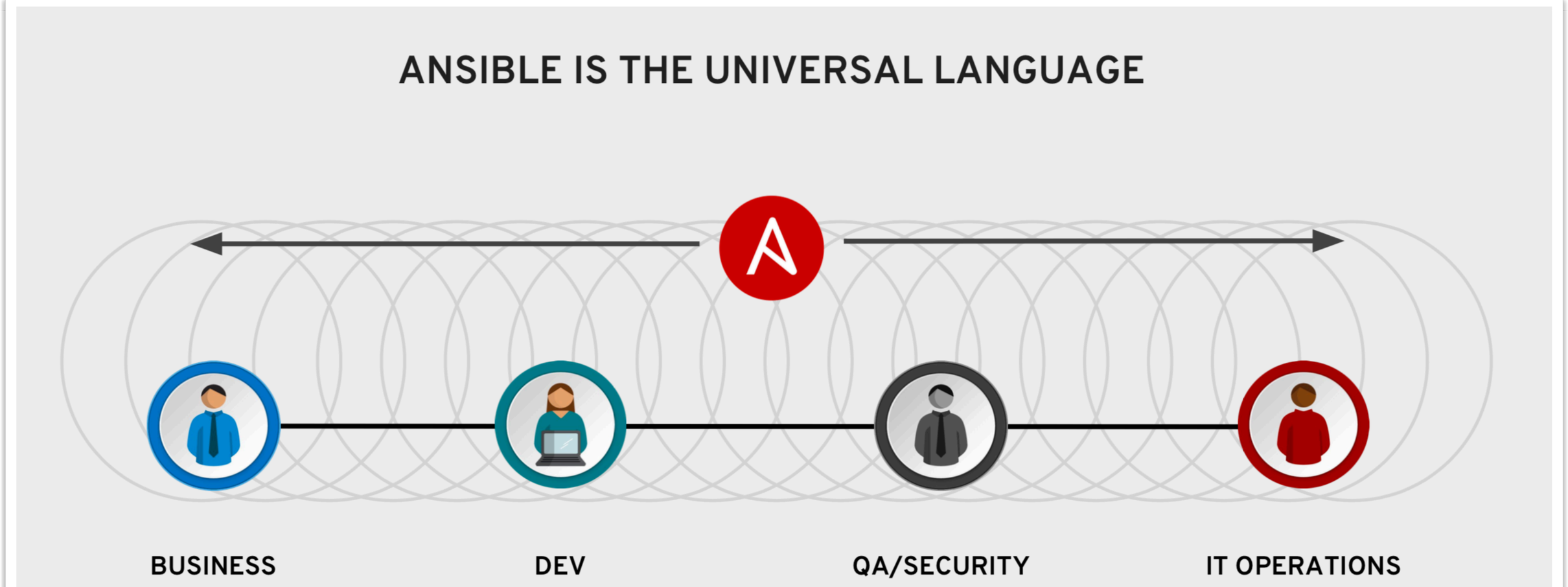
ANSIBLE WORKSHOP

Piotr Szewczuk

ANSIBLE & BUSINESS

ANSIBLE - ONE FOR ALL



Ansible is the first **automation language** that can be read and written across IT.

Ansible is the only **automation engine** that can automate the entire application lifecycle and continuous delivery pipeline.

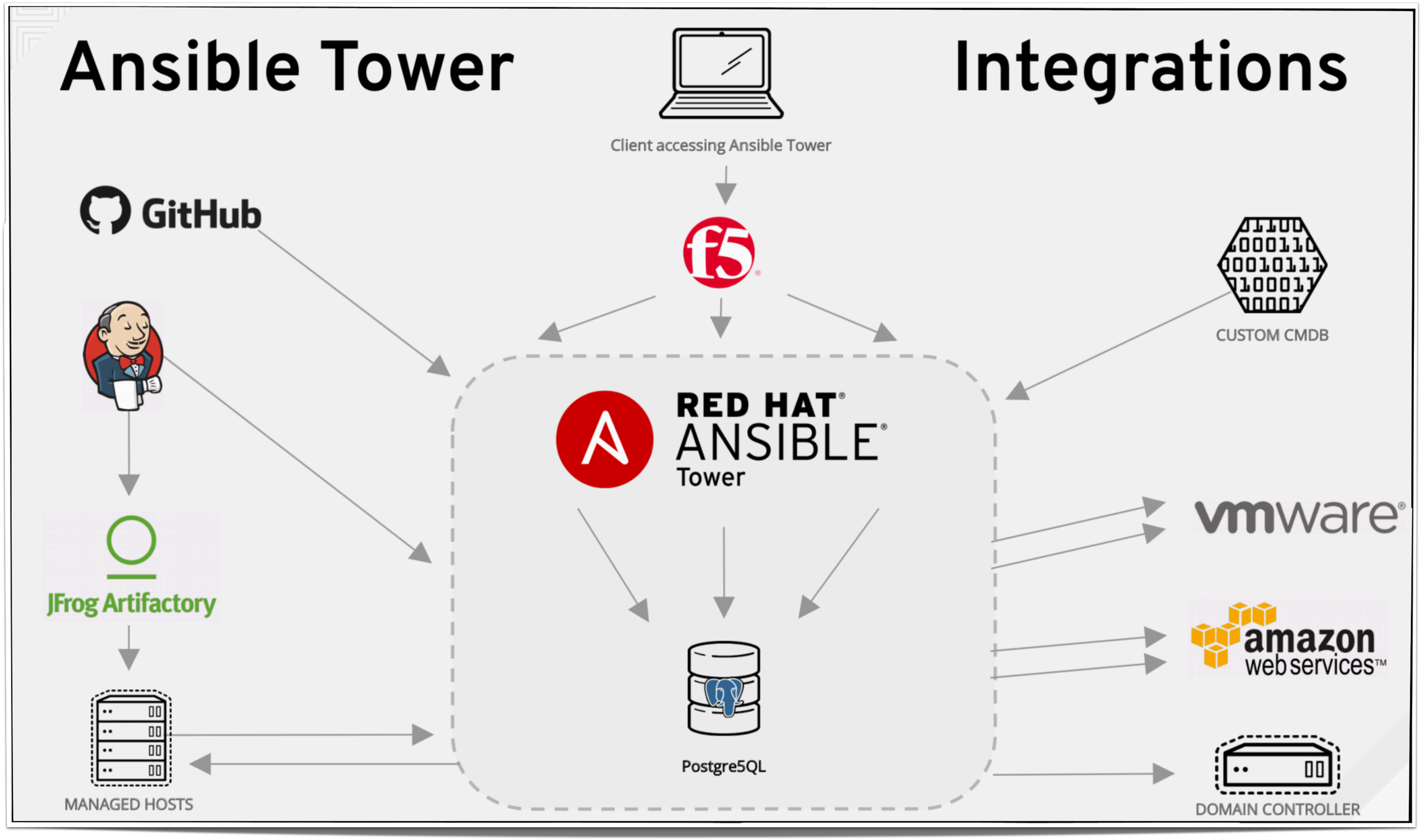
WHY ANSIBLE

- ✓ “Low cost” of entry into technologies
- ✓ Simplicity
- ✓ Agentless
- ✓ Idempotent
- ✓ Broad adoption
- ✓ Infrastructure as a code
- ✓ Config Management
- ✓ Provisioning
- ✓ Continuos Delivery & Continuos Integration - CI/CD
- ✓ Security & Compliance

ANSIBLE VS ...

	Puppet	Chef	Salt	Ansible
Wsparcie	Puppet Labs	Opscode	SaltStack	Red Hat
Technologia	Ruby	Ruby / Erlang	Python	Python
Komunikacja	SSL	SSL	ZeroMQ /SSH	SSH
Opis zadań	Manifest	Recipe	SLS Formula	Playbook
Komunikacja z serwerem master	Agent	Client	Minions	Agentless
Język konfiguracji	DSL	Ruby DSL	YAML	YAML
Wzorce	ERB	ERB	Jinja2	Jinja2
Dostępność	2005	2009	2011	2012

ANSIBLE - CI/CD



ANSIBLE INTEGRATION

✓ Virtualization: KVM, XEN, VMware, Hyper-V
✓ Operating Systems: Linux (Red Hat, CentOS, SUSE, Debian, Ubuntu), Windows, Unix
✓ Networks: A10, Arista, Cisco, Cumulus Networks, Dell, F5, Juniper, Open vSwitch
✓ Containers: Docker, LXC, OpenShift / OKD
✓ Cloud: OpenStack, CloudStack, AWS, Google Cloud, Microsoft Azure, Alibaba Cloud, Rackspace, Mirantis
✓ DevOps Tools: GitHub, GitLab, Vagrant, Jenkins, Travis CI
✓ Monitoring: Splunk, Dynatrace, InfluxDB, Sensu, Nagios
✓ Chat: Slack, HipChat, IRC

ANSIBLE INTRODUCTION

ANSIBLE

- ✓ Ansible: Open source configuration management and orchestration utility
- ✓ Automates and standardizes configuration of remote hosts and virtual machines
- ✓ Coordinates launch and shutdown of multitiered applications
- ✓ Performs rolling updates of multiple systems with zero downtime
- ✓ System administrators find it simple to use
- ✓ Developers can learn it easily
- ✓ Built on Python

ARCHITECTURE

- ✓ Two types of machines in Ansible architecture: control node and managed hosts
  - ✓ Ansible software installed and components maintained on control node
  - ✓ Managed hosts listed in host inventory

- ✓System administrators log in to control node and launch Ansible
  - ✓ Specify playbook
  - ✓ Specify target host to manage: single system, group of hosts, or wild card

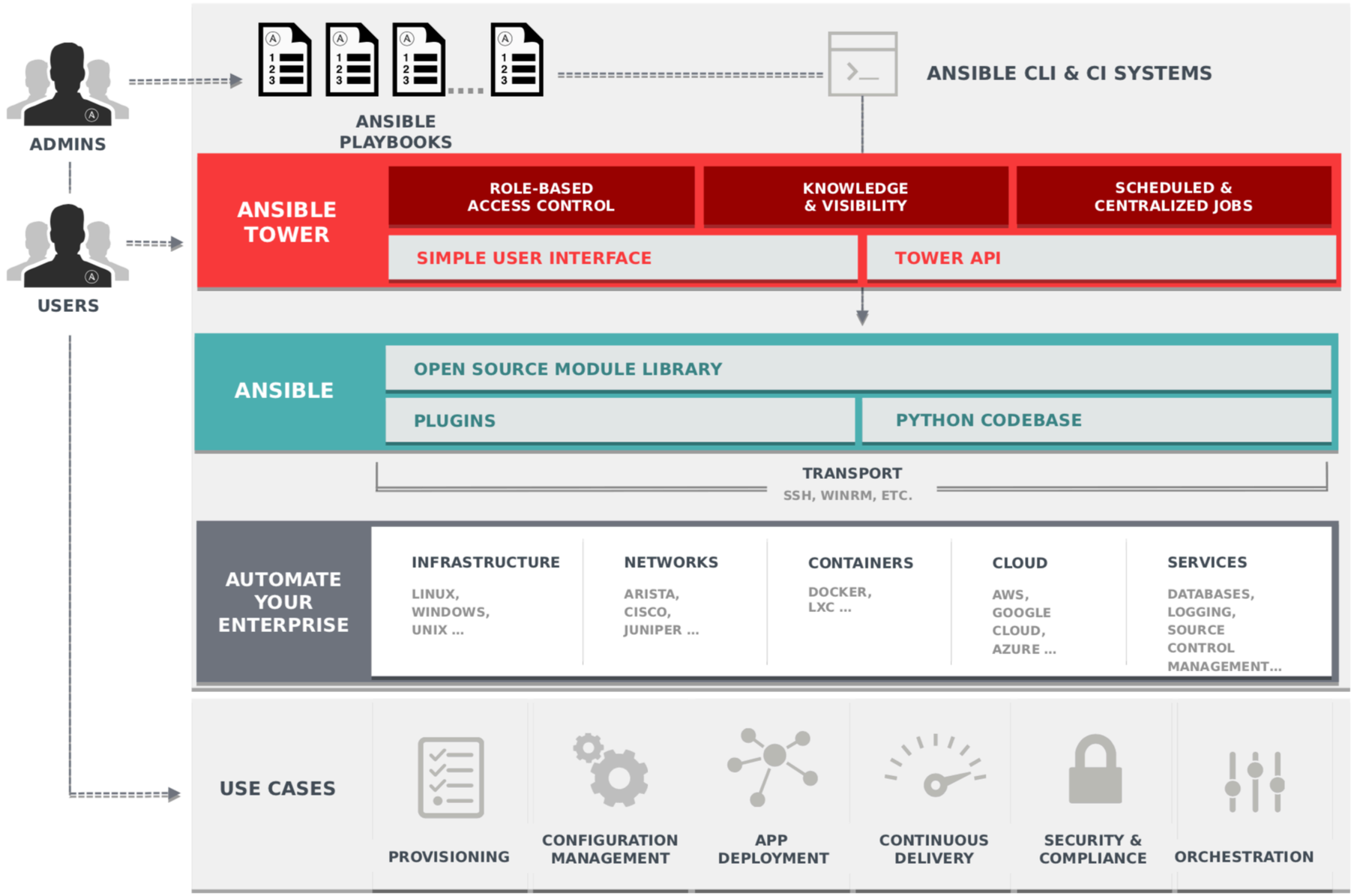
- ✓SSH used as network transport to communicate with managed hosts
  - ✓ Modules referenced in playbook copied to managed hosts
  - ✓ Modules execute in order with arguments specified in playbook

ANSIBLE TERMS

- ✓ Control node: the host on which you use Ansible to execute tasks on the managed nodes
- ✓ Managed node: a host that is configured by the control node
- ✓ Host inventory: a list of managed nodes
- ✓ Ad-hoc command: a simple one-off task
- ✓ Playbook: a set of repeatable tasks for more complex configurations
- ✓ Module: code that performs a particular common task such as adding a user, installing a package, etc.

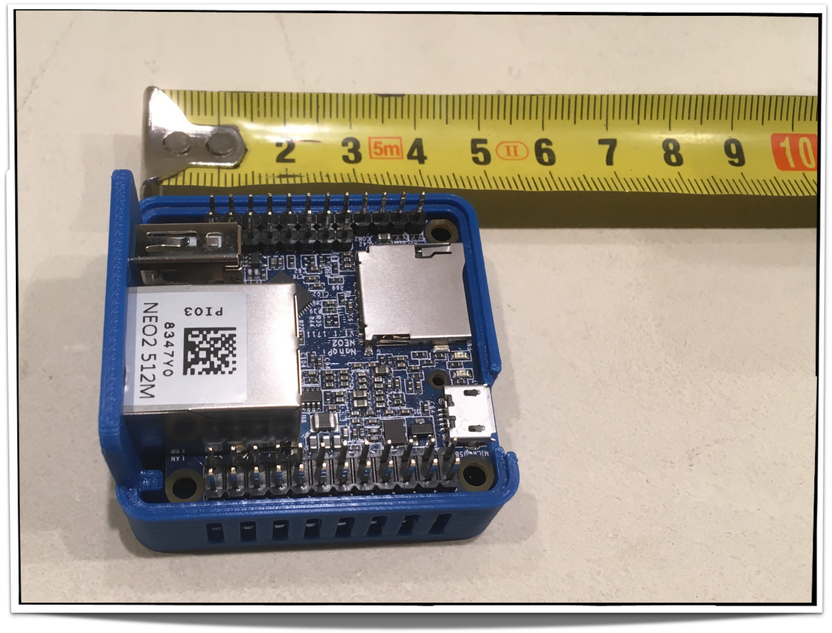
ANSIBLE STACK

PLATFORM OVERVIEW

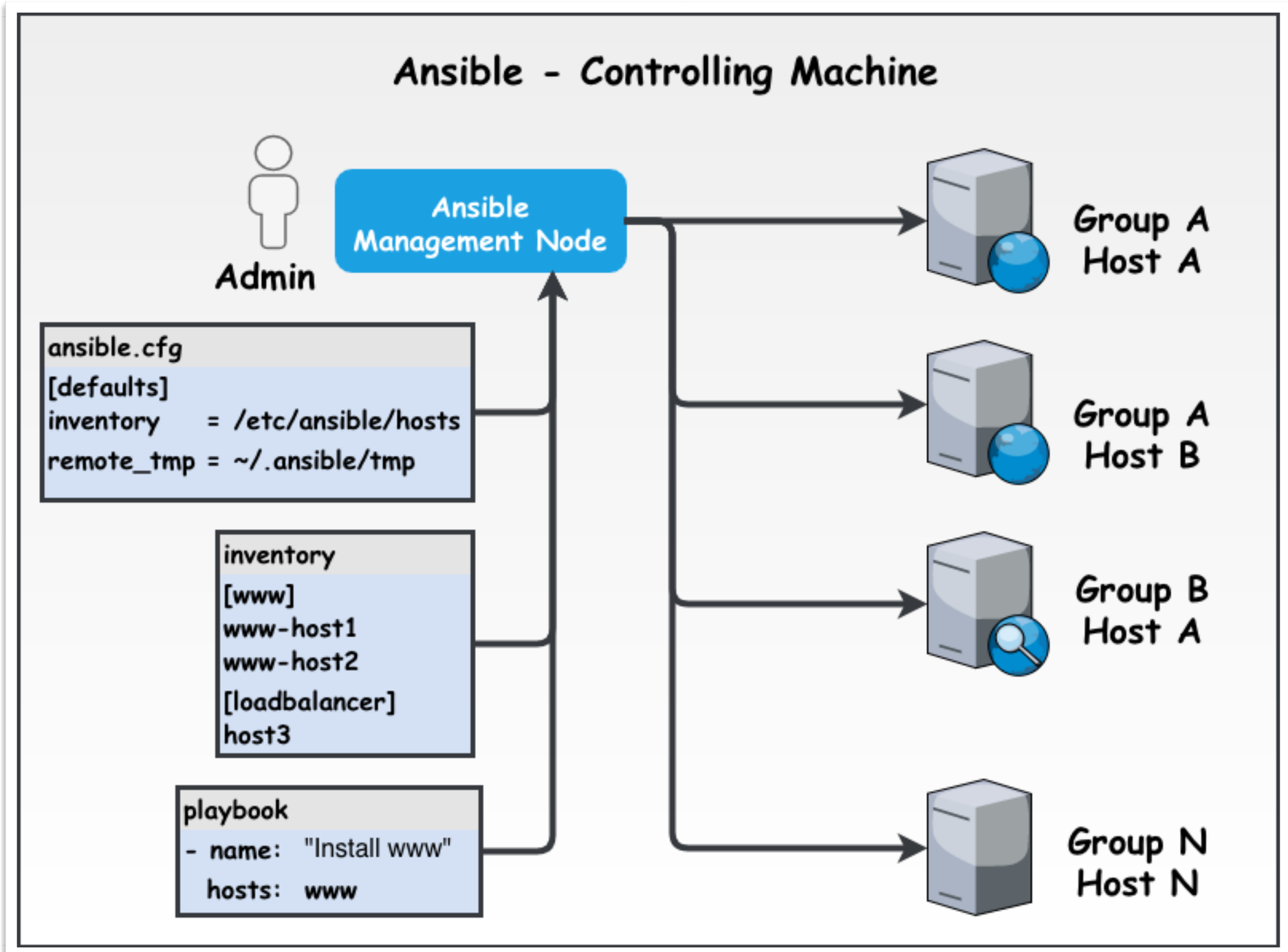


redhat.

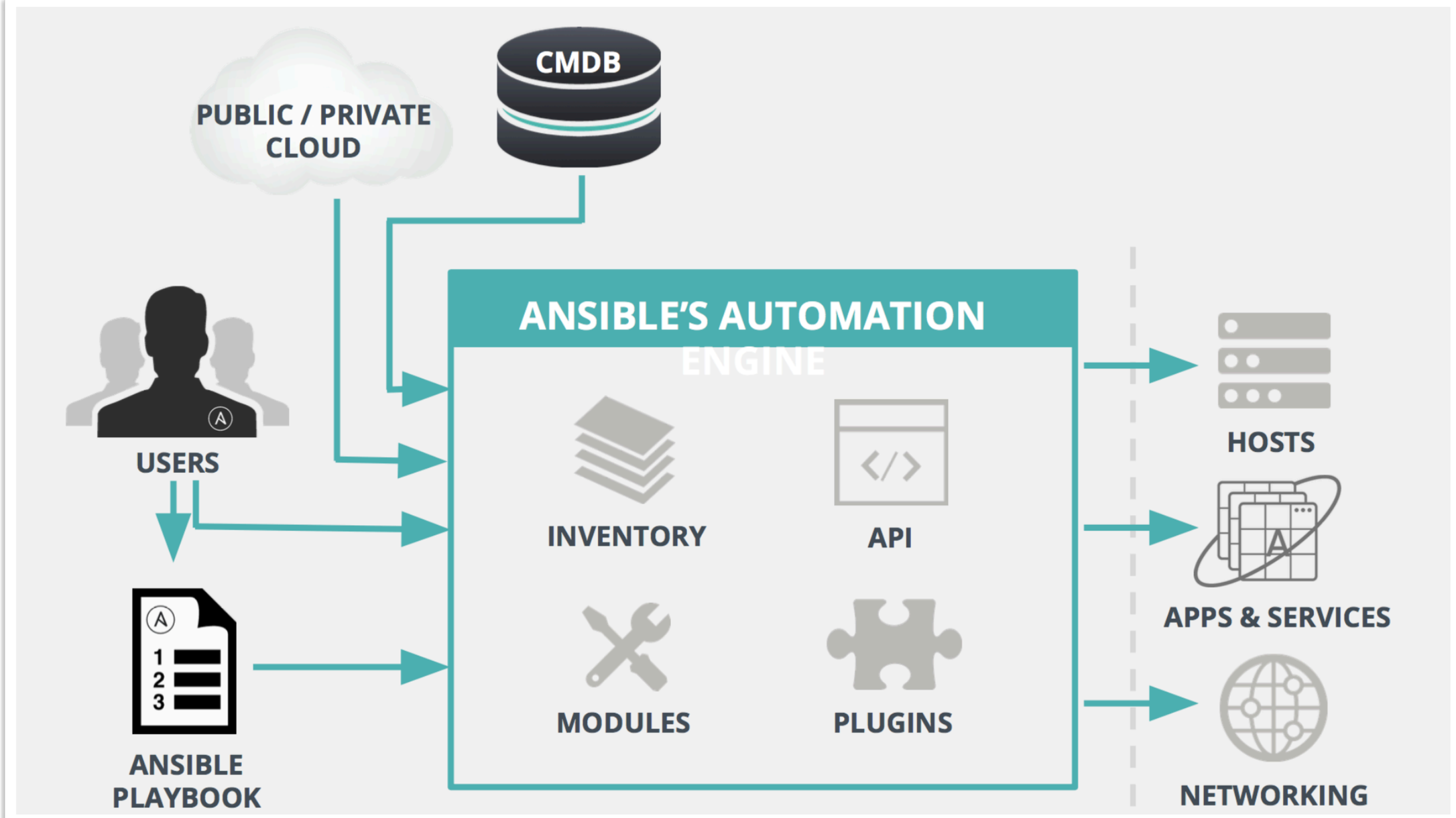
ANSIBLE CONTROL NODE



CONTROLLING MACHINES



ANSIBLE ARCHITECTURE



ANSIBLE DOC

- ✓ https://docs.ansible.com/ansible/latest

- ✓ ansible-doc -l
- ✓ ansible-doc yum

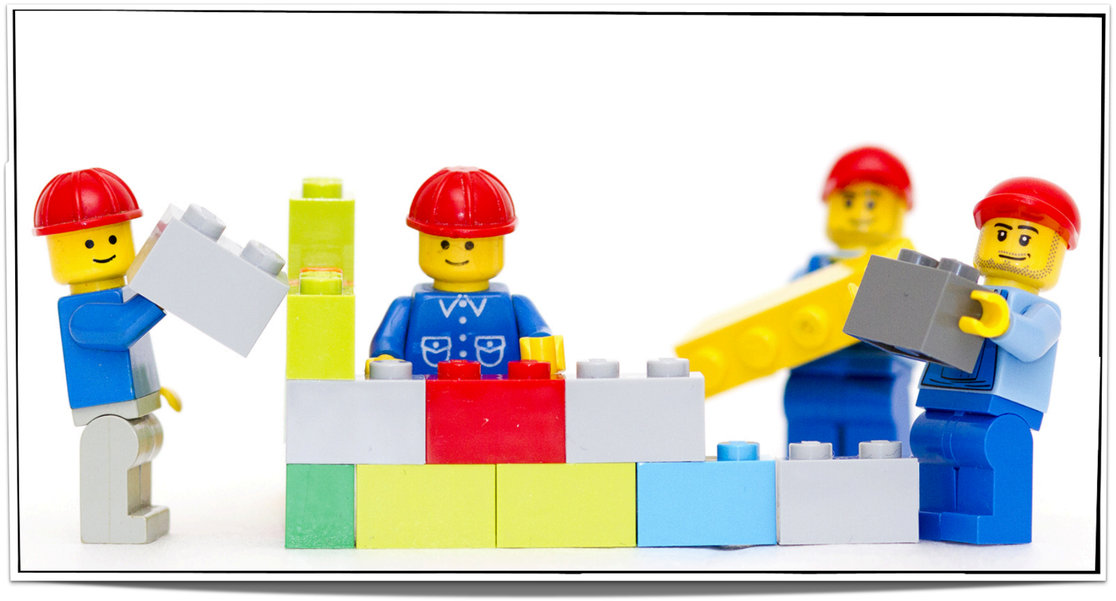
```
ansible --version | egrep '^[^ ]' && ansible-doc -l |wc -l

  ansible 2.9.21
  3387
```

ANSIBLE CONFIG

- ✓ **ansible-config** utility allows users to see all available configuration settings, their defaults, how to set them, and where current values come from
- ✓ Changes can be made in configuration file
- ✓ Ansible searches for file to use in this order:
  - ✓ ANSIBLE_CONFIG (environment variable, if set)
  - ✓ ansible.cfg (in current directory)
  - ✓ ~/.ansible.cfg (in home directory)
  - ✓ /etc/ansible/ansible.cfg
- ✓ First file found is used, all others ignored

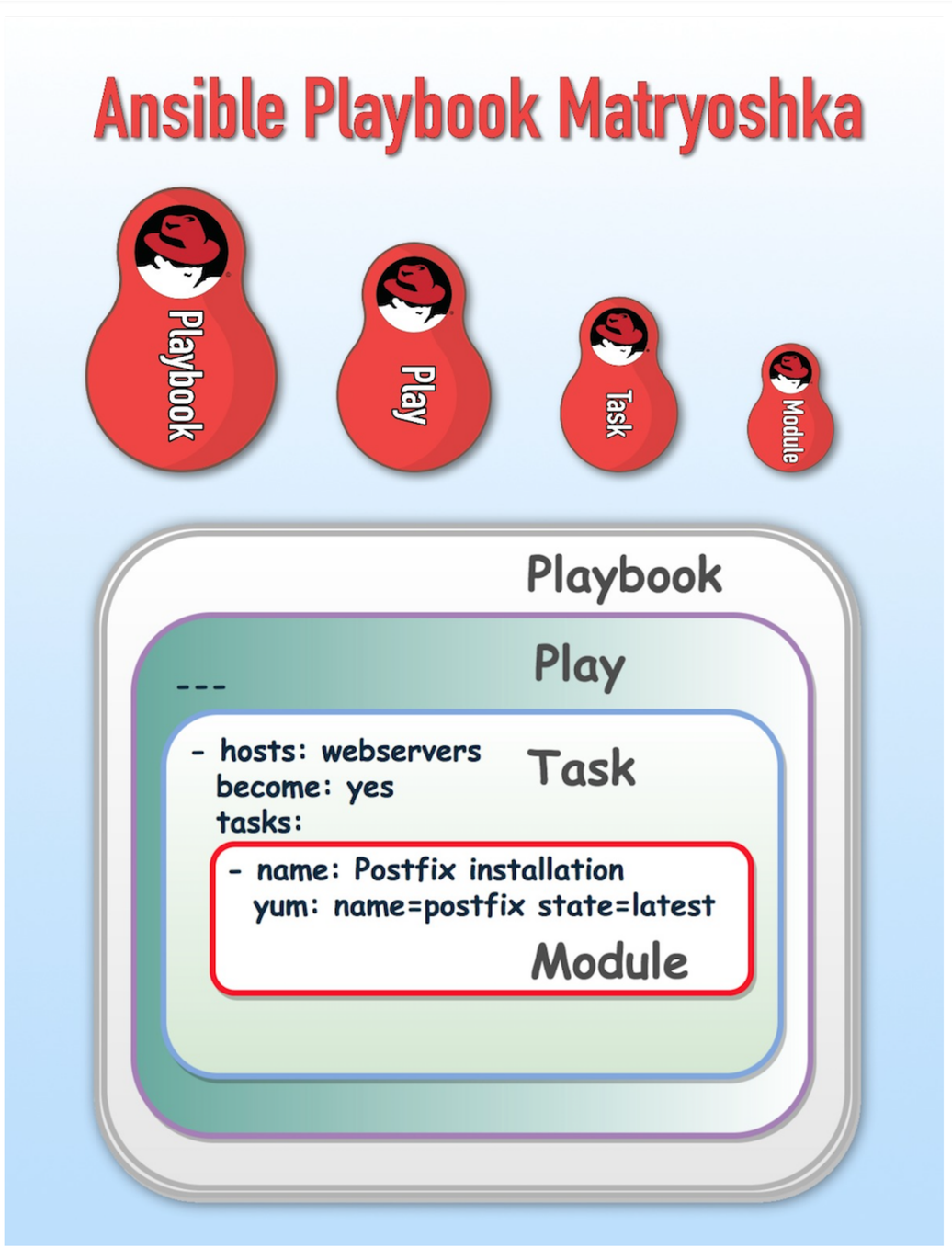
ANSIBLE BRICKS

- ✓ Inventory file
- ✓ Facts
- ✓ Modules
- ✓ Variables
- ✓ Loops
- ✓ Blocks
- ✓ Task Control
- ✓ Jinja2 Templates
- ✓ Ansible Vault
- ✓ Roles



PLAYBOOK

```
---
- name: install and start apache
  hosts: web
  become: yes
  vars:
    http_port: 80

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

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

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



INVENTORY

```
[defaults]
inventory = /path/inventory

[www]
node1 loadbalancer_node_ip=192.168.10.12 loadbalancer_node_name=node1
node2 loadbalancer_node_ip=192.168.10.14 loadbalancer_node_name=node2

[loadbalancer]
node3

[db:children]
loadbalancer
```

ROLES

```
[root@instalator instalator]# ansible-galaxy init rola
- rola was created successfully
[root@instalator instalator]# tree ./rola/
./rola/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

YAML

✓ YAML - YAML Ain't Markup Language - https://yaml.org/
✓ YAML Rules:
- ➡ Indentation
- ➡ Colons
- ➡ Dashes

✓ YAML validator:
- ➡ http://www.yamllint.com/
- ➡ http://yaml-online-parser.appspot.com/

YAML RULES

Indentation - YAML uses a fixed indentation scheme to represent relationships between data layers.

vim ~/.vimrc

syntax on
set ts=2 sw=2
set cursorcolumn
set number
set paste

YAML RULES

Colons - Dictionary keys are represented in YAML as strings terminated by a trailing colon. Values are represented by either a string following the colon, separated by a space.

```
my_key: my_value

first_level_dict_key:
  second_level_dict_key: value_in_second_level_dict
```

YAML RULES

Dashes - To represent lists of items, a single dash followed by a space is used. Multiple items are a part of the same list as a function of their having the same level of indentation.

```
- list_value_one
- list_value_two
- list_value_three
```

PLAYBOOK RUN

✓ [vagrant@master devops]$ **ansible-playbook** 10-tomcat-install.yml **- -step**

✓ [vagrant@master devops]$ **ansible-playbook** 10-tomcat-install.yml **- -start-at-task=“Install Java”**

✓ [vagrant@master devops]$ **ansible-playbook** 10-tomcat-install.yml **- -syntax-check**

✓ [vagrant@master devops]$ **ansible-playbook** 10-tomcat-install.yml **- -check**

✓ [vagrant@master devops]$ **ansible-playbook** 10-tomcat-install.yml **- -check -diff** <— reports the changes done to the template files on managed hosts

LOCAL ANSIBLE.CFG

[defaults]

inventory = inventory

callback_whitelist = profile_tasks
 # Use the YAML callback plugin.
stdout_callback = yaml
 # Use the stdout_callback when running ad-hoc commands.
bin_ansible_callbacks = True

roles_path = roles

ANSIBLE LOG

✓ by default Ansible is not configured to log its output to any log file

✓ $ANSIBLE_LOG_PATH env variable

✓ log_path in ansible.cfg

[defaults]
log_path = /home/vagrant/devops

FILES MODULES

✓ acl - Sets and retrieves file ACL information.
✓ archive - Creates a compressed archive of one or more files or trees
✓ assemble - Assembles a configuration file from fragments
✓ blockinfile - Insert/update/remove a text block surrounded by marker lines
✓ copy - Copies files to remote locations
✓ fetch - Fetches a file from remote nodes
✓ file - Sets attributes of files
✓ find - Return a list of files based on specific criteria
✓ ini_file - Tweak settings in INI files
✓ lineinfile - Manage lines in text files
✓ replace - Replace all instances of a particular string in a file using a back-referenced regular expression.
✓ stat - Retrieve file or file system status
✓ template - Templates a file out to a remote server
✓ unarchive - Unpacks an archive after (optionally) copying it from the local machine.
✓ xattr - Manage user defined extended attributes
✓ xml - Manage bits and pieces of XML files or strings