Ansible:
a Software Architecture Analysis

Presented by:
Simone Berni

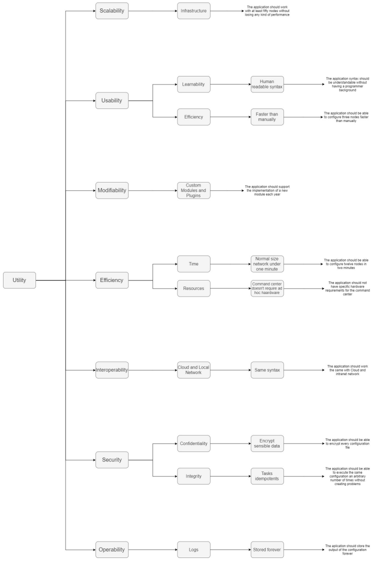
December 29, 2019

What is Ansible?

Ansible is an open-source IT automation engine.
Its main tasks are:

- Hosts orchestration
- Application deployment
- Configuration management
- Cloud Provisioning

Utility tree
Non functional requirements


Utility
Scalability
Infrastructure
The application should work with at least fifty nodes without losing any kind of performance
Usability
Learnability
Human readable syntax
The application syntax should be understandable without having a programmer background
Efficiency
Faster than manually
The application should be able to configure three nodes faster than manually
Modifiability
Custom Modules and Plugins
The application should support the implementation of a new module each year
Efficiency
Time
Normal size network under one minute
The application should be able to configure twelve nodes in two minutes
Resources
Command center doesn't require ad hoc haardware
The application should not have specific hardware requirements for the command center
Interoperability
Cloud and Local Network
Same syntax
The application should work the same with Cloud and intranet network
Security
Confidentiality
Encrypt sensible data
The application should be able to encrypt every configiuration file
Integrity
Tasks idempotents
The application should be able to execute the same configuration an arbitrary number of times without creating problems
Operability
Logs
Stored forever
The aplication should store the output of the configuration forever

Use cases

Use Case	Description	Scenario
UC1	Manage Users	Every node of the infrastructure has the same set of users and every user is able to log in every machine.
UC2	Install Application	Every node of the infrastructure should be able to install the same software, even if the machines have different OS, packet manager and configurations.
UC3	Provide reports	Whenever the nodes in the network are modified via Ansible, a report of what has be done, what was sucessfull and what not, has to be created.
UC4	History and Downgrade	For each node of the infrastructure the administrator has to be able to see its history and has to be able to downgrade the system to a previus version.

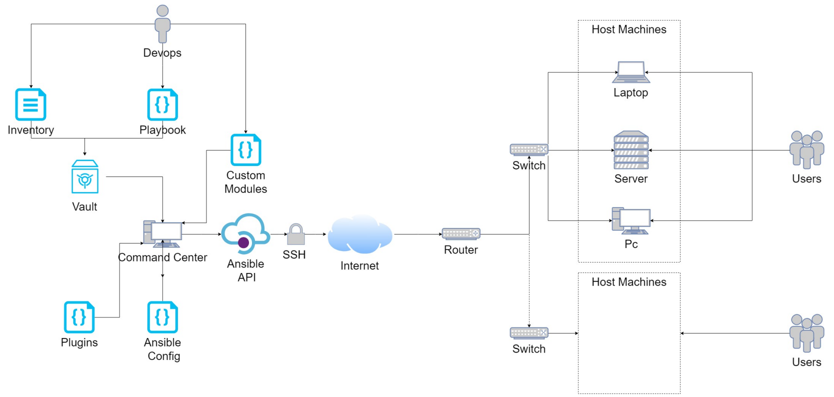
Use Case	Description	Scenario
UC5	Configuration Syntax	The syntax has to be easy to remember and to understand, since the administrator wants to focus on what has to be done, not how to write it.
UC6	Manage Cloud	The user should be able to use Anisble to manage many Amazon AWS instances.
UC7	Extendable	If the builtin modules doesn't answer a user problem, should be easy to create and integrate a new custom module.
UC8	Secure Connection	The sensible data should travel in a secure communication channel to the nodes.

Non functional requirements

Id	Requirement	Scenario	Associated Use Cases
NF1	Usability	The application should be able to configure three nodes easier and faster than manually	UC1
NF2	Usability	The application should be able to configure any part of the system using the same syntax structure	UC1, UC2, UC5
NF3	Modifiability	The application should allow the creation of any kind of custom modules	UC7
NF4	Efficiency	The application should be able to configure a medium size net-work(15 nodes) in under 2 minute	UC6
NF5	Interoperability	The application should work the same with Cloud and company's network	UC6

Id	Requirement	Scenario	Associated Use Cases
NF6	Security	The application should be able to connect to the hosts via SSH	UC8
NF7	Security	The application should be able to encrypt sensible data in a secure way	UC8
NF8	Operability	The application should store the history for each node forever	UC4
NF9	Scalability	The application should work with at least 50 nodes without losing performance	UC2, UC6

Structures



The Playbook is like an instruction manual, where the Inventory is the material, the Modules are the tools and the Vault the cabinet

Structures
Inventory

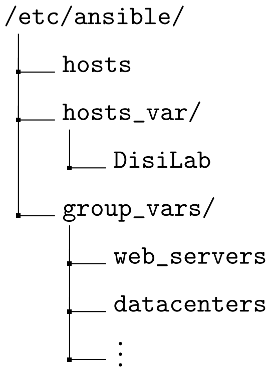
- List or groups of hosts that Ansible will work against
- Can be used to assign value to variables and to create aliases
- Is possible to create a filesystem grouping many inventory files

Structures
Inventory - Example

```
[dbservers]
db01.test.example.com
db02.test.example.com

[appservers]
app01.test.example.com
app02.test.example.com
app03.test.example.com
```

Inventory example



Inventory filesystem example

Structures
Module

- A small program that is executed on each host
- It describes the desired state of the system
- Has to be idempotent
- The output can be *ok*, *changed*, *failed*
- It is used inside a Playbook
- Is possible to create custom modules

Structures

Module - Example

```
name: restart webserver
    service:
      name: httpd
      state: restarted
```

Module service example

```
name: write apache config
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf
```

Module template example

Structures
Vault

- Manages the encryption and decryption of sensible data
- Can encrypt every Ansible file
- Is possible to have variable-level encrpytion
- The unlocking password can be inserted via prompt or password manager

Structures
Playbook

- Is a list of play, or tasks, that must be done against some hosts
- Enable the possibility to logically group tasks
- Yaml Syntax
- Must be idempotent
- Supports *Jinja2* templating
- Enable the possibility to retrieve *facts*
- Supports conditionals and loops

Structures
Playbook - Example

```
---
hosts: webservers
remote_user: root
tasks:
  name: apache latest
  yum:
    name: httpd
    state: latest
  name: apache config
  template:
    src: /srv/httpd.j2
    dest: /etc/httpd.conf
```

Playbook example

```
---
hosts: webservers
remote_user: root
tasks:
  name: apache latest
  yum:
    name: httpd
    state: latest
  name: apache config
  template:
    src: "{{local_path}}httpd.j2"
    dest: "{{remote_path}}httpd.conf"
```

Playbook jinja2 example

Structures
Plugin

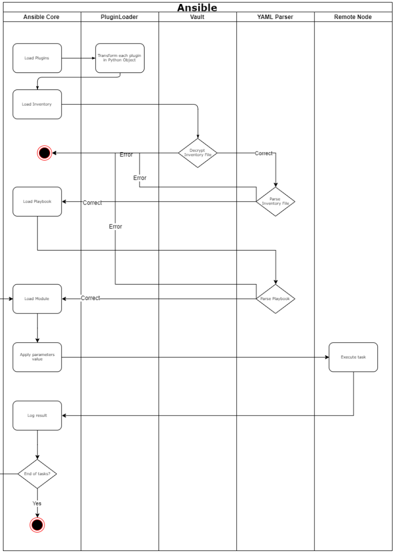
- Pieces of code that augment Ansible’s core functionality
- The code is normally executed on the control machine, not on the hosts
- Many are shipped with Ansible
- Is possible to create custom plugins
- Example: Connection plugin

Functions

Req	Component	Description
Usability Efficiency	Module	Ansible offers enough modules to complete every task that the user need to accomplish on the remote machines with just one Yaml line.
Usability Syntax	Playbook	Ansible uses Yaml syntax, and each play has the same structure and syntax, but different keys and values.
Modifiability	Module and Plugin	Ansible offers the possibility to create new modules and plugins that must return JSON data.
Scalability	Inventory and Playbook	Using Ansible's inventory is possible to group hosts together and to run the same task to every host sequentially.

Req	Component	Description
Interoperability	Module	For the devops engineer, Ansible works exactly the same with Cloud and local hosts, but has to use differents modules to accomplish the same result.
Security Connection	Module	SSH is the default way to execute task.
Security Encryption	Vault	The vault utility offers the possibility to encrypt files without reducity the Ansible power.
Operability History	Plugin	The plugin *log_plays* will save the Playbook log to a file, but is a devops engineer task to preserve them in a solid way.

Behaviour
Work flow



Critical Analysis
Pros

- **Easy to learn**: clear documentation full of examples
- **Agentless**: fast deploy on every node
- **Yaml syntax**: human readable, without having to remember a complex keywords or an ad hoc syntax
- **Opensource**: the source code is available to everyone
- **Python**: the fastest growing language nowadays

Critical Analysis
Cons

- **CLI only**: designed to be configurable from its prefered text editor, an attempted has beed made with Ansible Tower
- **Young**: is the most recently developed orchestration tool, and inevitably bugs could occur
- **SSH**: modules execution relie completely on SSH

Similar Architectures

- **Chef**
- **Puppet**
- **Saltstack**

Similar Architectures
Chef

- Architecture: **Master-agent**
- Fault tollerance: **Backup server**
- Language: **Ruby DSL**
- OS: **Master Unix, Clients whatever**

Similar Architectures
Puppet

- Architecture: **Master-agent**
- Fault tollerance: **Multi master**
- Language: **Puppet DSL**
- OS: **Master Unix, Clients whatever**

Similar Architectures
OpenStack

- Architecture: **Master-agent**
- Fault tollerance: **Can be Multi master**
- Language: **YAML**
- OS: **Master Unix, Clients whatever**

Conclusions

Ansible is built with a proper design in mind: no overhead. For this reason it doesn’t support daemons, it uses YAML, the execution is via SSH and is only configurable using the prefered text editor.
This may create some issues, but the pros outweight the cons.

References

P.Masek, M. Stusek, J. Krejci, K. Zeman, J. Pokorny, M. Kudlacek, ''Unleashing Full Potential of Ansible Framework: University Labs Administration'', IEEE, Jyvaskyla, Finland, 5-18 May 2018, [2018 22nd Conference of Open Innovations Association (FRUCT)].
W. Yiran, Z. Tongyang, G. Yidong, ''Design and implementation of continuous integration scheme based on Jenkins and Ansible'', IEEE, Chengdu, China, 26-28 May 2018, [2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)].
N. K. Singh, S. Thakur, H. Chaurasiya, H. Nagdev, ''Automated provisioning of application in IAAS cloud using Ansible configuration management'', IEEE, Dehradun, India, 4-5 Sept. 2015, [2015 1st International Conference on Next Generation Computing Technologies (NGCT)].
V. Shvetcova, O. Borisenko, M. Polischuk,''Domain-Specific Language for Infrastructure as Code'', IEEE, Velikiy Novgorod, Russia, 13-14 Sept. 2019, [2019 Ivannikov Memorial Workshop (IVMEM)].
J. Keating, ''Mastering Ansible'', Packt Publishing Ltd., Birmingham, Nov. 2015, pp 1-33.
"Ansible Documentation", Accessed on: Nov. 25, 2019, [Online], Avaible: https://docs.ansible.com/ansible/latest/index.html