# Ansible

Reporter **Andranik Barseghyan**

https://github.com/Rebiss/Presentation

# Process Diagram

**What is Ansible**

**Why Ansible**

**Ansible Use Cases**

**Example**

**Configuration Management
With Ansible**

**Architecture of Ansible**
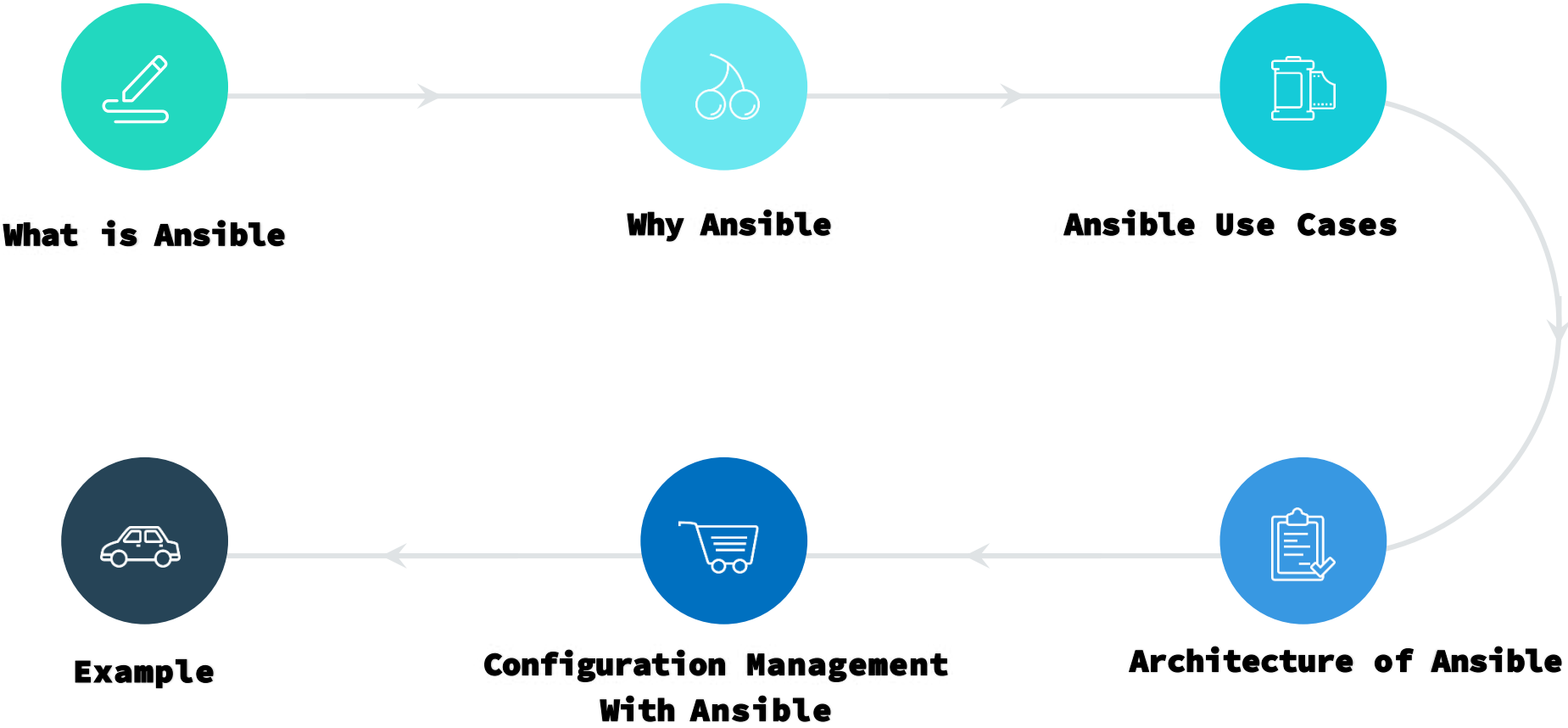
# What is Ansible

● Ansible is an open-source configuration management and provisioning tool, similar to Chef, Puppet or Salt

● It uses SSH to connect to servers and run the configured  Tasks. Ansible lets you control and configure nodes from a  single machine

● What makes it different from other management software is  that Ansible uses SSH infrastructure. The project was founded in 2013 and bought by Red Hat in 2015.

**Free**. Ansible is an open-source tool.

Very simple to set up and use

Powerful. Ansible lets you model even highly complex IT workflows.

Flexible and Agentless

Efficient

# Why Ansible

**No Agent**

As long as the box can be ssh'd into and it has python, it can be configured with Ansible.

Other configuration tools tend to be procedural — do this and then do that and so on. Ansible works by you writing a description of the state of the machine that you want and then it takes steps to fulfill that description.

**Declarative Not Procedural**

**Idempotent**

Ansible's whole architecture is structured around the concept of idempotency. The core idea here is that you only do things if they are needed and that things are repeatable without side effects.

Ansible is quite easy to learn. It doesn't require any extra knowledge.

**Tiny Learning Curve**
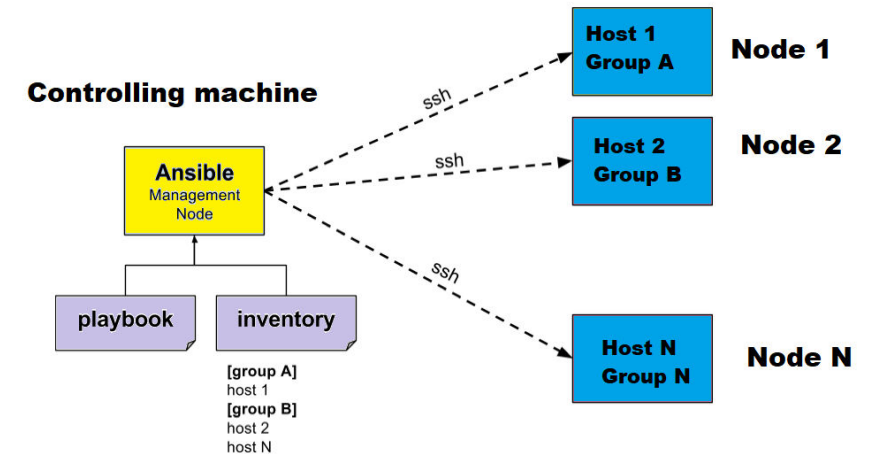
# Difference

Push | Pull
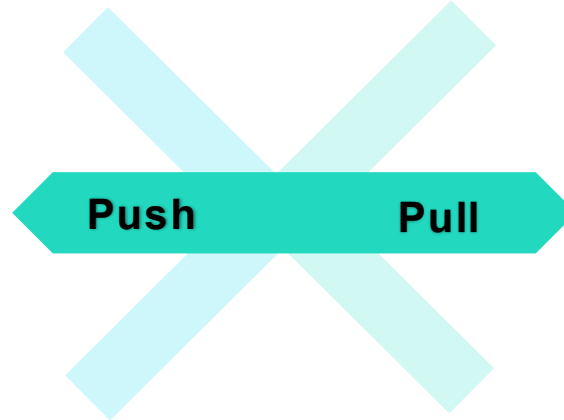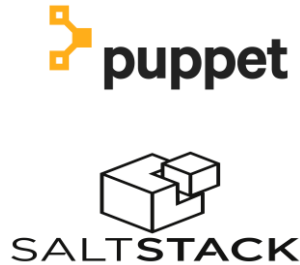
**Controler Server - Master Server**

- Linux Only (RedHat, Debian, CentOS, OS X)
- Python 2.6+ or Python 3.5+

**Controlled Server - Managed Server**

- Linux: Root Username/Password or SSH Key Python 2.6+. SSH port 22
- Windows: Admin Username/Password, PowerShell 3.0 (ConfigureRemotingForAnsible.ps1). WinRM Port 5986



Controlling machine

Ansible
Management
Node

playbook | inventory

[group A]
host 1
[group B]
host 2
host N

Host 1
Group A → Node 1

ssh

Host 2
Group B → Node 2

ssh

ssh

Host N
Group N → Node N

# Ansible Use Cases

**Weekly system reboot**

There's nothing worse than doing the same thing for 8 hours a day! Eliminate repetitive, manual processes with automation.

**Enforce security guidelines**

Rules are rules. It's best to automate in an effort to achieve strict security standards.

**Monitor configuration drift**

Use check mode with Ansible tasks to enforce desired settings and see if your configuration has drifted.

**Disaster recovery**

Disaster recovery can involve a wide range of components. Act across different variables of the technology stack to identify problems and eliminate cross team dependencies.

**Command blaster**

Remarkably easy to write, you can run commands across your environment for any number of servers.
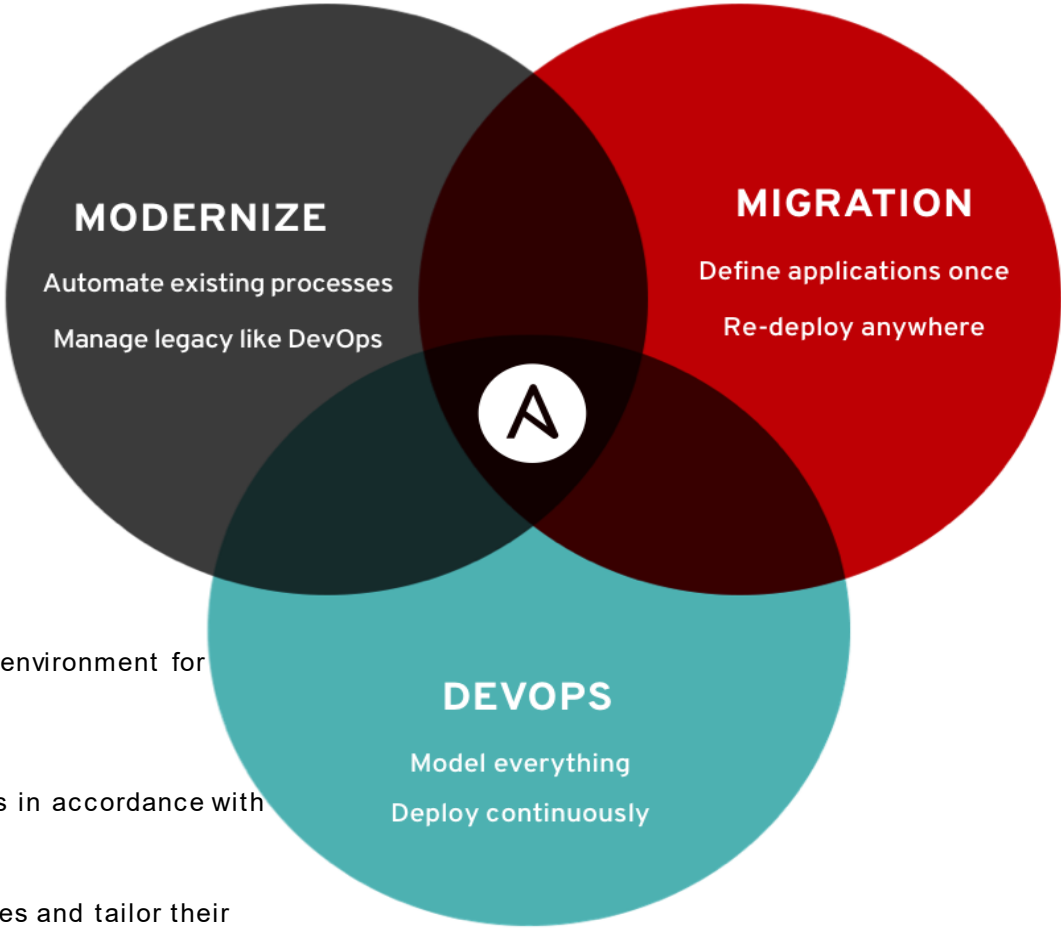
**Database binary patching**

Several databases use outdated binary sets. Patch the binaries in accordance with the release of the latest patch.
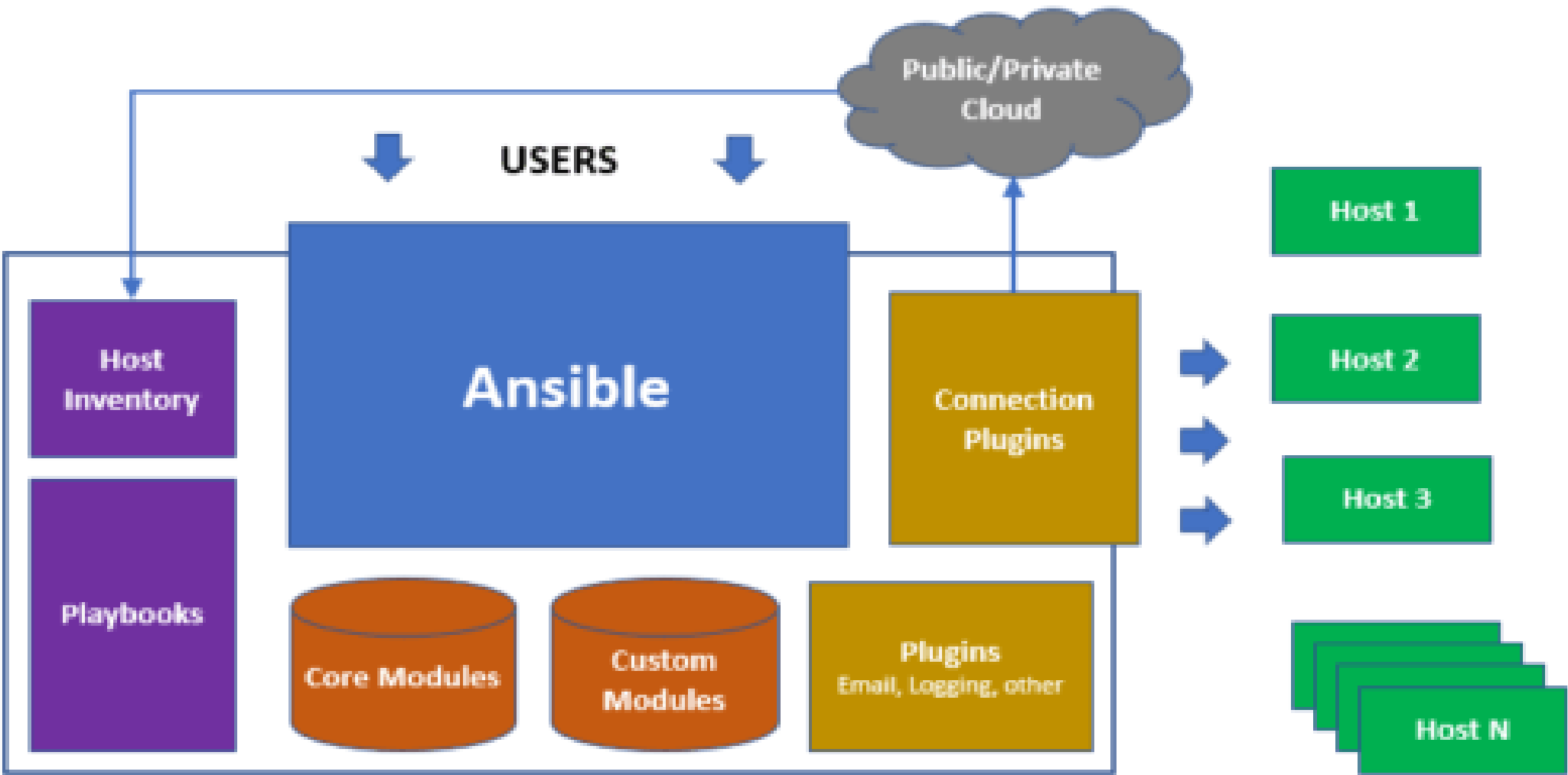
**Instance provisioning**

Use modules for several cloud providers to create new instances and tailor their configuration.

**Service license agreements**

Mistakes cost time and money. Eliminate errors that can crop up in detailed software contracts.

**MODERNIZE**

Automate existing processes

Manage legacy like DevOps

**MIGRATION**

Define applications once

Re-deploy anywhere

**DEVOPS**

Model everything

Deploy continuously

# Architecture of Ansible

## Host Inventory

The Inventory is a description of the nodes that can be accessed by Ansible. By default, the Inventory is described by a configuration file, whose default location is in./etc/ansible/hosts The configuration file lists either the IP address or hostname of each node that is accessible by Ansible. Every host is assigned to a group such as web servers, db servers etc. The inventory file can be in one of many formats such as yaml, INI etc
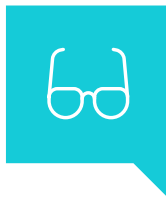
## Playbook

Playbooks are simple YAML files. These files are descriptions of the desired state of your systems. Ansible then does the hard work of getting your systems to that state no matter what state they are currently in. Playbooks make your installations, upgrades and day-to-day management repeatable and reliable. Playbooks are simple to write and maintain. Playbooks are written in a natural language so they are very easy to evolve and edit.

.

## Modules

There are over 1000 modules provided by Ansible to automate every part of the environment. Modules are like plugins that do the actual work in Ansible, they are what gets executed in each playbook task. Each module is mostly standalone and can be written in a standard scripting language (such as Python, Perl, Ruby, Bash, etc.). One of the guiding properties of modules is idempotency, which means that even if an operation is repeated multiple times, it will always place the system into the same state.

## Roles

Roles are a way to group tasks together into one container. We could have a role for setting up MySQL, another one for configuring iptables etc. Roles makes it easy to configure hosts. Any role can be performed on any host or group of hosts such as:

# Configuration Management With Ansible

Ansible is the simplest solution for configuring the nodes. It's designed  to be minimal in nature, consistent, secure and highly reliable. Any  developer, tester or IT manager can easily configure nodes. Any IT  person can write playbooks easily. Ansible configurations are simple data descriptions of your infrastructure (human readable) ensuring everyone on your team will be  able to understand the meaning of each configuration task. Ansible requires nothing more than a password or SSH key in order to  start managing systems and can start managing them without installing  any agent software.
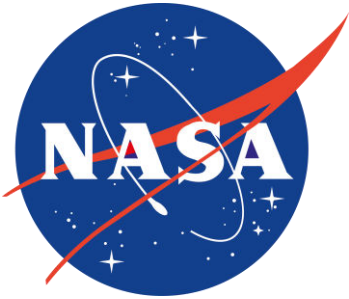
**Example**

# References

## Companies that are using Ansible

✓ www.digitalocean.com/community/tutorials/configuration-management-101-writing-ansible-playbooks
✓ www.ansible.com
✓ docs.ansible.com
✓ docs.ansible.com/ansible/latest/installation_guide/intro_configuration.html

Thank You