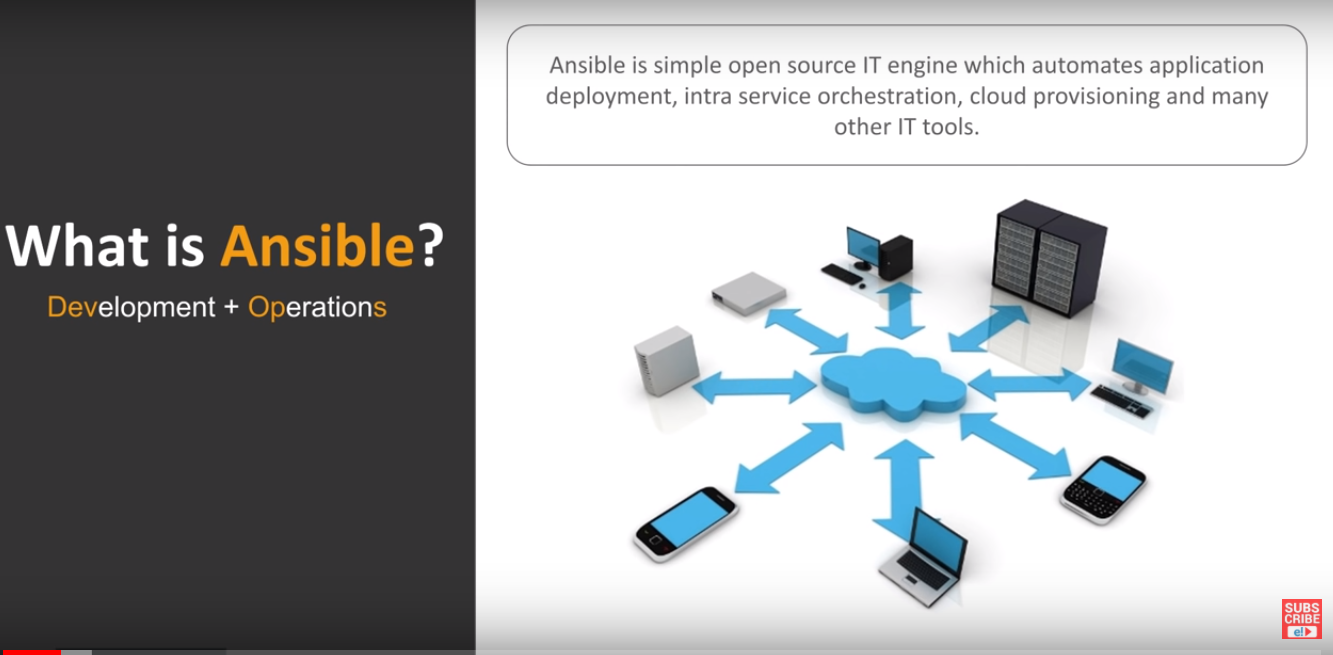
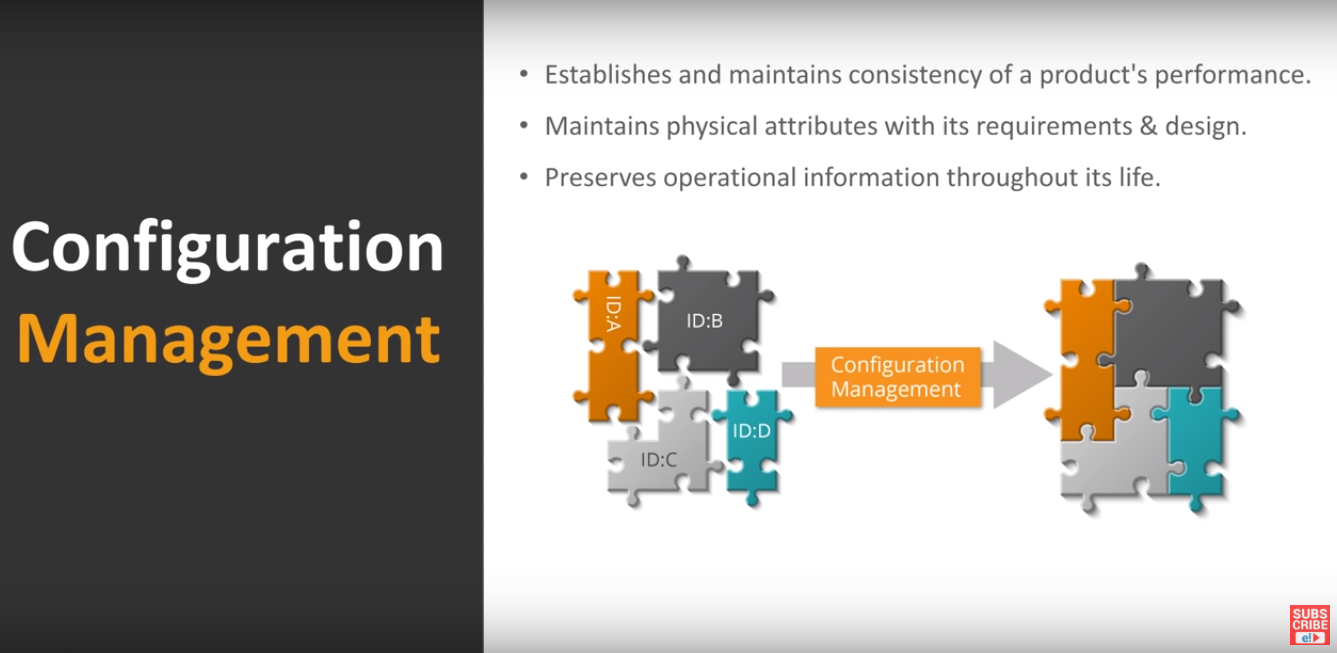
Ansible

Ansible is an IT automation tool. It can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments or zero downtime rolling updates.

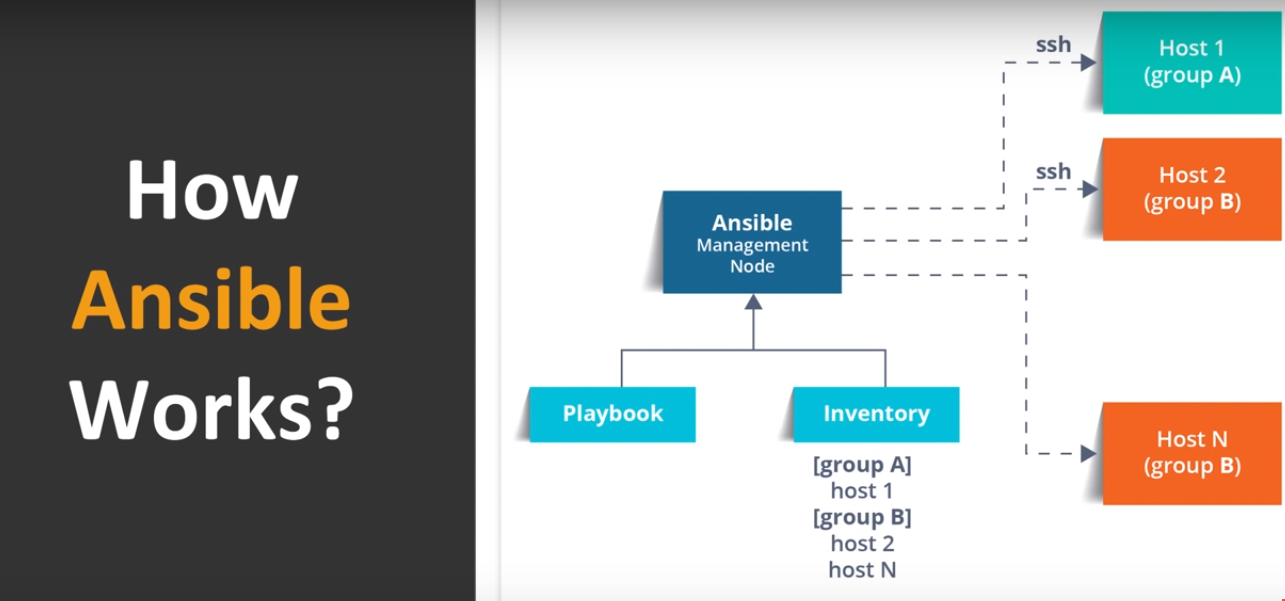
Ansible’s main goals are simplicity and ease-of-use. It also has a strong focus on security and reliability, featuring a minimum of moving parts, usage of OpenSSH for transport (with other transports and pull modes as alternatives), and a language that is designed around auditability by humans–even those not familiar with the program.

We believe simplicity is relevant to all sizes of environments, so we design for busy users of all types: developers, sysadmins, release engineers, IT managers, and everyone in between. Ansible is appropriate for managing all environments, from small setups with a handful of instances to enterprise environments with many thousands of instances.



Ansible works against multiple systems in your infrastructure at the same time. It does this by selecting portions of systems listed in Ansible’s inventory, which defaults to being saved in the location /etc/ansible/hosts. You can specify a different inventory file using the -i option on the command line.

Not only is this inventory configurable, but you can also use multiple inventory files at the same time and pull inventory from dynamic or cloud sources or different formats (YAML, ini, etc).



Ansible Architecture

Ansible is a radically simple IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs.

Being designed for multi-tier deployments since day one, Ansible models your IT infrastructure by describing how all of your systems inter-relate, rather than just managing one system at a time. It uses no agents and no additional custom security infrastructure, so it’s easy to deploy - and most importantly, it uses a very simple language (YAML, in the form of Ansible Playbooks) that allow you to describe your automation jobs in a way that approaches plain English.

Modules

Ansible works by connecting to your nodes and pushing out small programs, called “Ansible Modules” to them. These programs are written to be resource models of the desired state of the system. Ansible then executes these modules (over SSH by default), and removes them when finished.

Your library of modules can reside on any machine, and there are no servers, daemons, or databases required. Typically you’ll work with your favorite terminal program, a text editor, and probably a version control system to keep track of changes to your content.

Plugins

Plugins are pieces of code that augment Ansible’s core functionality. Ansible ships with a number of handy plugins, and you can easily write your own.

Inventory

By default, Ansible represents what machines it manages using a very simple INI file that puts all of your managed machines in groups of your own choosing. To add new machines, there is no additional SSL signing server involved, so there’s never any hassle deciding why a particular machine didn’t get linked up due to obscure NTP or DNS issues.

If there’s another source of truth in your infrastructure, Ansible can also plugin to that, such as drawing inventory, group, and variable information from sources like EC2, Rackspace, OpenStack, and more.

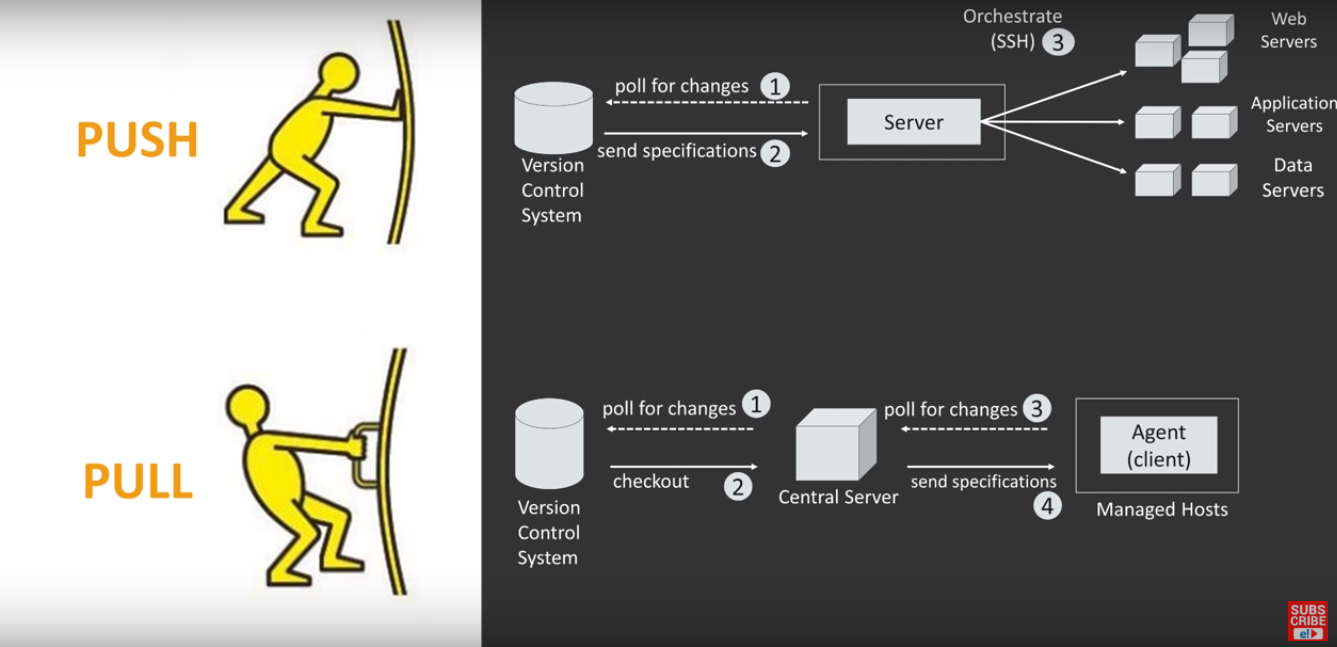
Playbooks

Playbooks can finely orchestrate multiple slices of your infrastructure topology, with very detailed control over how many machines to tackle at a time. This is where Ansible starts to get most interesting. Ansible’s approach to orchestration is one of finely-tuned simplicity, as we believe your automation code should make perfect sense to you years down the road and there should be very little to remember about special syntax or features

Hosts and Groups

The inventory file can be in one of many formats, depending on the inventory plugins you have. For this example, the format for /etc/ansible/hosts is an INI-like (one of Ansible’s defaults) and looks like this: 

The headings in brackets are group names, which are used in classifying systems and deciding what systems you are controlling at what times and for what purpose.



Playbooks can declare configurations, but they can also orchestrate steps of any manual ordered process, even as different steps must bounce back and forth between sets of machines in particular orders. They can launch tasks synchronously or asynchronously. While you might run the main /usr/bin/ansible program for ad-hoc tasks, playbooks are more likely to be kept in source control and used to push out your configuration or assure the configurations of your remote systems are in spec.

Ansible-Pull

Should you want to invert the architecture of Ansible, so that nodes check in to a central location, instead of pushing configuration out to them, you can. The ansible-pull is a small script that will checkout a repo of configuration instructions from git, and then run ansible-playbook against that content. Assuming you load balance your checkout location, ansible-pull scales essentially infinitely. Run ansible-pull --help for details. There’s also a clever playbook available to configure ansible-pull via a crontab from push mode.