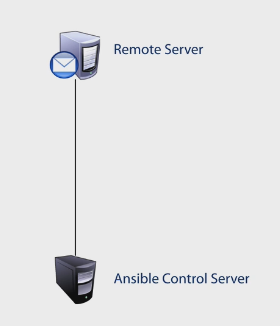
**Hands-on Ansible**

<https://www.pluralsight.com/courses/hands-on-ansible>

**01 - Architecture and Process Flow**

****

* **Requirements**:
  + Ansible Control Server
    - Python 2.6+
    - Must be \*Nix environment (Linux, Unix, Mac)
    - Windows not supported.
  + Remote Server
    - You can manage switchers, firewalls, router, load balancers, etc.
    - Python 2.5+ (Simplejson) - **Python 3.x Is not supported**
    - SSH
    - \*NIX
    - Ansible can support windows machines: Remote PowerShell Enabled.

**Component overview**

**Inventory**:

* It is a text file where you describe your servers and your systems.
  + You define your **host variables, groups, roles, user accounts** to access certain systems.

**Modules:**

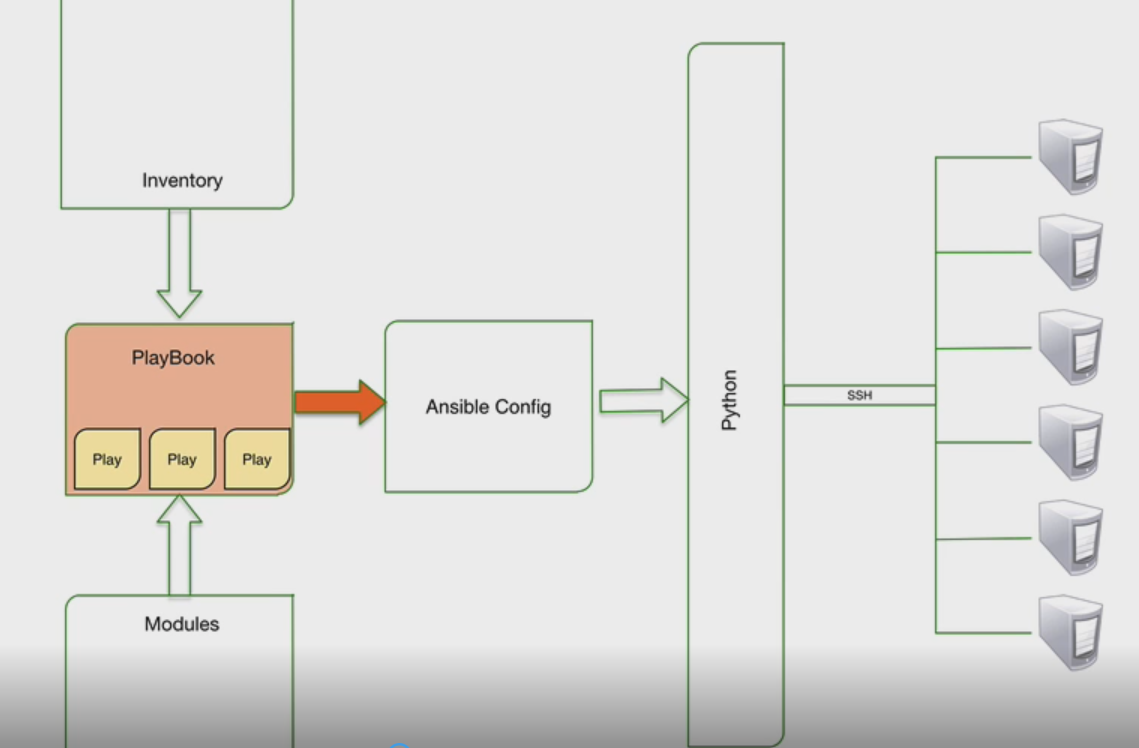
* + A programmed unit of work to be done.
  + Ansible galaxy: Community based catalogue
    - **yum module for RedHat**
    - **apt module for Debian**
      * You tell the module what you want to do.

**Playbook**

- YAML Files you are going to create to perform a task or a set of Tasks.

- Plays will be executed in a predefined order.

- A **play** can a single or a set of **tasks** using modules, executed on a defined set of hosts.



- A **Playbook** is a grouping of individual plays, built in a specific order sequence to produce an expected outcome or outcomes across many different set of hosts.

**Example**: Using a playbook, you can:

* + Install an application
  + Copy over a configuration file
  + Switch over the DB servers.
  + Configure the DB.
* You can also add **logic** to the **playbook**: Checking if the drive has free space to install before running the install.
* Check for the version of the OS.
* Etc.

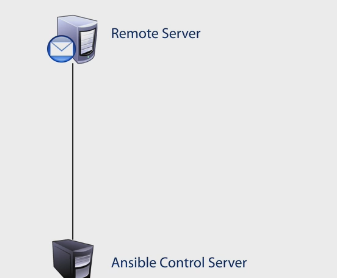
**Ansible configuration file:**

* Global Text file:
  + How many parallel operations your system can perform at the same time?

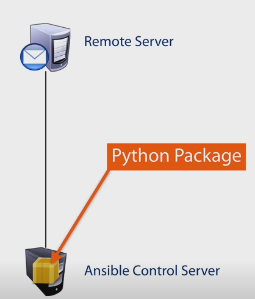
**Python**

* Building variables for automation and orchestration.
* **Host variables:** 
  + Use variables defined in Inventory per host or group.
* **Facts variables**
  + Whenever you run a play, it can gather data from that system like memory usage, @Ip addresses, CPU speed.
* **Dynamic variables:**
  + Created during a playbook and destroyed
  + Created based on the result of a play and use it in another play.

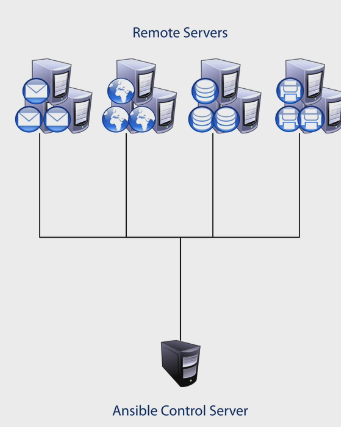
**Process of execution and Flow**

****

* Process in order to talk to a remote server.
* Evaluate the playbook that you have created.
  + Playbook Identifies the system to deploy to and the modules to grab
  + Once everything is identified, it creates a **python package.**

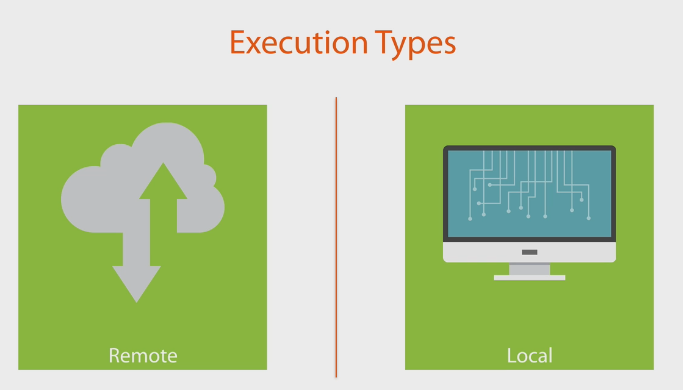


* After it has identified the system to deploy to, using the **Inventory module,** it is going to establish an **SSH session and deploy the package according to the config parameters given.**
* Copy a package to a tmp directory on a remote machine.
* Python on the remote machine will execute the package and gives you results via a JSON on which steps were successful.
* Result will be turned back to Ansible using JSON and report of status.
* The tmp package will be deleted on the remote system, keeping everything clean.
* Ansible will move on to the next play on that playbook.



* Last case was from a system to a one system
* It can target Mail server, file server, DB server…all at the same time.

**Executions types**

****

**Remote execution:**

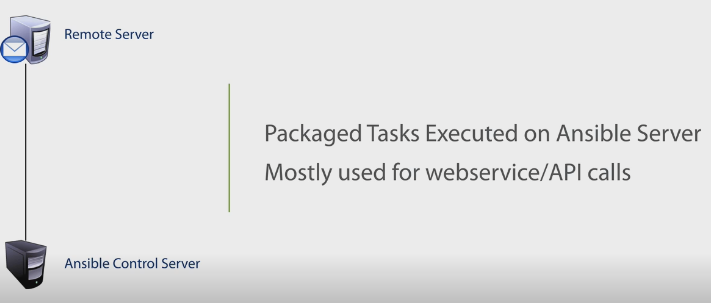
1 – Python package is delivered to remote system

****

2- Package will be executed there and result JSON will be returned



**Local execution:**

****

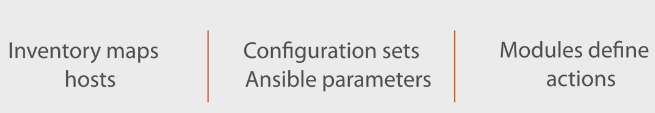
**You don’t need to send a package but just do a RESTful call to a remote machine and get the HTTP codes, then run the package locally.**

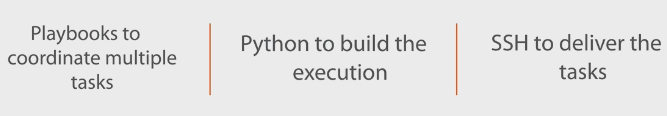
**Architecture conclusion**

* **Inventory component**:

Maps our hosts to Ansible. We define our remote systems here, grouping them together in categories. There can be variable that a host may need.

* **Configuration Sets Ansible params.**
* **Modules components:** Call the modules to execute the actions

****

****