Slide 2:

**What is Ansible:**

- Born in 2012

- Written in Python

- Maintained by RedHat

- Current ver 2.1.1.0

- GNU License

**Configuration Management:**

Provisioning: Traditionally in Networking/Telecommunication

- Server Provisioning: select server, OS, Middleware, Drivers, etc.

Can be either baremetal or VM

Security: Low attack surface area

**Ziploader (Ansible 2.1+):** compiles common code into a zip file (with a wrapper script) which is then extracted and imported on the remote system.

module\_utils/basic.py is used by all modules which are shipped with Ansible, and defines the AnsibleModule class. The class handles most of the low-level details required of modules (arg parsing and checking, atomic file operation, etc.)

**Slide 7:**

**Company-wide abstraction:**

Treating Infra as code has lotf of advantages. But comes with costs. Not everybody is willing to learn everything. Few powerful people with Ansible will become bottleneck. A module to abstract away as much as possible, has a clear behavior and a navigable definition.

This provides an easy interface that can be shared across the company.

Module writing strategies: There are 3 major ways you can write a module:

- Wrap a CLI command

Pro:

easy to write

Cons:

Output has to be scraped out of the CLI output. Is error prone.

Localization

- Use a 3rd party library (most common)

Pros:

Again very easy

All the hard work is done by the author of that library

Cons:

Adding dependency to your module

Adds and extra layer of difficulty- you will have to use python request everywhere you run the module

Bugs and abandonment of the library

- Interact with the API directly (most powerful one)

Pros:

No extra dependencies (Assuming it’s REST API)

New features are added immediately

Cons:

Having to know the API