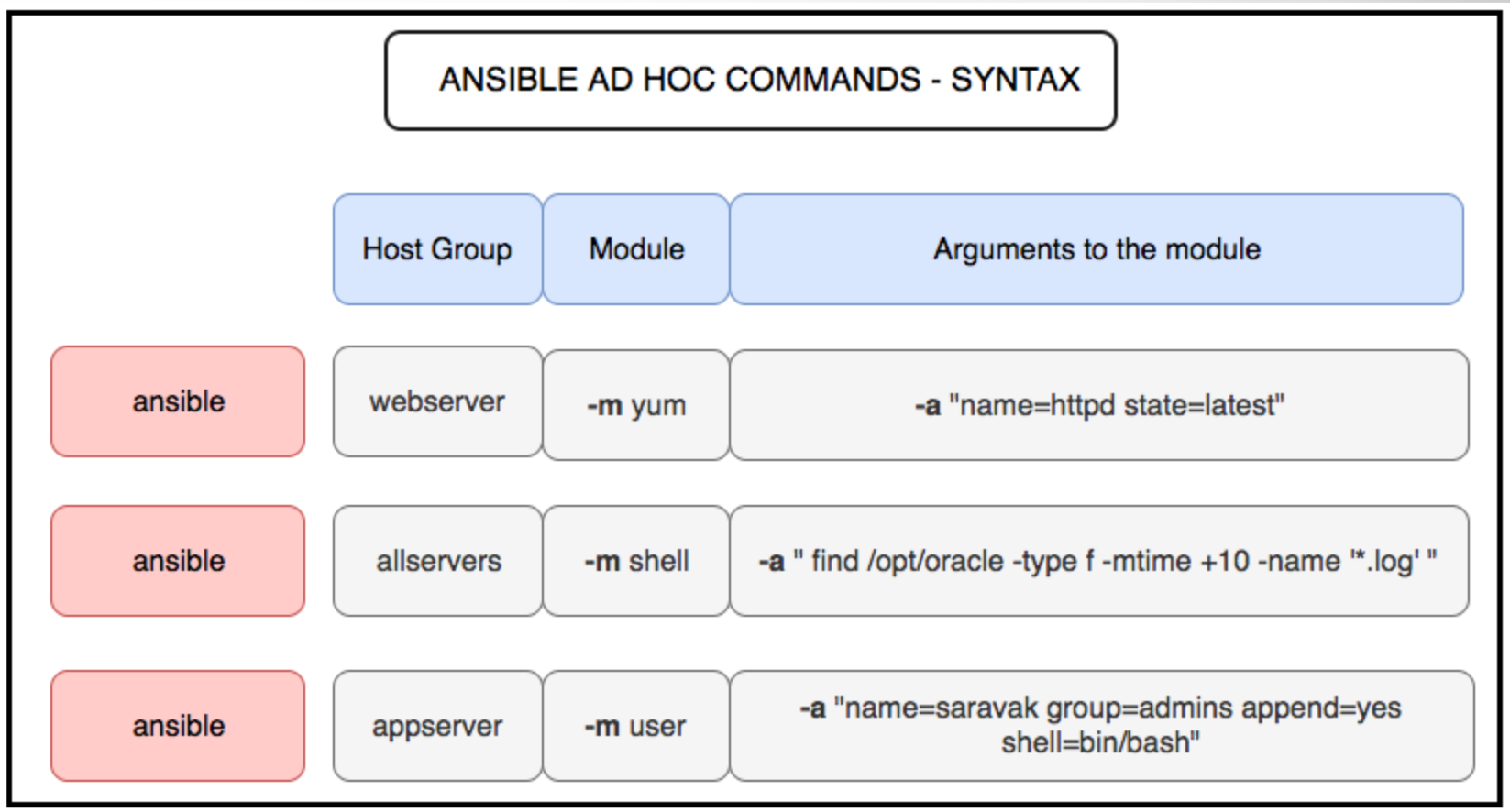
Ansible Ad-Hoc Commands

What's an ad-hoc command?

- Ansible ad hoc commands are CLI commands used for simple and one-time tasks.
- An ad-hoc command is something that you might type in to do something really quick, but don't want to save for later.

Ansible ad_hoc commands Syntax



Parallelism and Shell Commands (1-2)

- Let's use Ansible's command line tool to reboot all web servers in Atlanta, 10 at a time.
- Now to run the command on all servers in a group, in this case, Atlanta, in 10 parallel forks:

```
$ ansible atlanta -a "/sbin/reboot" -f 10
```

Parallelism and Shell Commands (2-2)

- /usr/bin/ansible will default to running from your user account. If you do not like this behavior, pass in "-u username".
- If you want to run commands as a different user, it looks like this:

```
$ ansible atlanta -a "/sbin/reboot" -f 10 -u username
```

Gathering Facts

- Facts are described in the playbooks section and represent discovered variables about a system.
- These can be used to implement conditional execution of tasks but also just to get ad-hoc information about your system. You can see all facts via:

```
$ ansible all m setup
```

Performing Tasks with Modules Using Ad Hoc Commands

- **Modules** are the tools that **ad hoc commands** use to accomplish tasks.
- Ansible provides **hundreds of modules** which do different things.
- find a **tested**, **special-purpose** module that does what you need as part of the standard installation.
- The **ansible-doc -l** command lists all modules installed on a system.
- You can use **ansibledoc** to view the documentation of particular modules by name, and find information about what arguments the modules take as options.

The following table lists a number of useful modules as examples. Many others exist.

Module category	Modules
Files modules	• copy: Copy a local file to the managed host • file: Set permissions and other properties of files • lineinfile: Ensure a particular line is or is not in a file • synchronize: Synchronize content using rsync
Software package	• package: Manage packages using autodetected package manager native to the operating system • yum: Manage packages using the YUM package manager • apt: Manage packages using the APT package manager • dnf: Manage packages using the DNF package manager • gem: Manage Ruby gems • pip: Manage Python packages from PyPI
modules	• firewalld: Manage arbitrary ports and services using firewalld • reboot: Reboot a machine • service: Manage services • user: Add, remove, and manage user accounts
Net Tools modules	• get_url: Download files over HTTP, HTTPS, or FTP • nmcli: Manage networking • uri: Interact with web services

Idempotent modules

- Most modules are **idempotent**, which means that they can be run **safely multiple times**, and if the system is already in the **correct state**, they **do nothing**. For example, if you run the previous ad hoc command again, it should report **no change**:

```
[root@ansimaster ~]# ansible -m user -a 'name=newbie uid=4000 state=present' \
>servera.lab.example.com
servera.lab.example.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "comment": "",
    "createhome": true,
    "group": 4000,
    "home": "/home/newbie",
    "name": "newbie",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 4000
}
```

Time Limited Background Operations

- Long running operations can be run in the background, and it is possible to check their status later.
- For example, to execute long_running_operation asynchronously in the background, with a timeout of 3600 seconds (-B), and without polling (-P):

```
$ ansible all -B 3600 -P 0 -a "usr/binlong_running_operation --do-stuff"
```

- If you do decide you want to check on the job status later, you can use the async_status module, passing it the job id that was returned when you ran the original job in the background:

```
$ ansible web1.example.com async_status -a "jid=488359678239.2844"
```