Ad Hoc Command

- An ad-hoc command is something that you might type in to do something really quick, but don't want to save for later.
- This is a good place to start to understand the basics of what Ansible can do prior to learning the playbooks language – ad-hoc commands can also be used to do quick things that you might not necessarily want to write a full playbook for.
- Generally speaking, the true power of Ansible lies in playbooks. Why would you use ad-hoc tasks versus playbooks?
- For instance, if you wanted to power off all of your lab for Christmas vacation, you could execute a quick one-liner in Ansible without writing a playbook.

Ansible Modules

- Modules (also referred to as “task plugins” or “library plugins”) are discrete units of code that can be used from the command line or in a playbook task.
- Each module supports taking arguments. Nearly all modules take key=value arguments, space delimited.
- Some modules take no arguments, and the command/shell modules simply take the string of the command you want to run.
- Modules should be idempotent, and should avoid making any changes if they detect that the current state matches the desired final state.
- When using Ansible playbooks, these modules can trigger ‘change events’ in the form of notifying ‘handlers’ to run additional tasks.

Ansible Inventory

- 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 <path> option on the command line.
- 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:

Ansible Playbooks

- Playbooks are expressed in YAML format (see YAML Syntax) and have a minimum of syntax, which intentionally tries to not be a programming language or script, but rather a model of a configuration or a process.
- Each playbook is composed of one or more ‘plays’ in a list.
- The goal of a play is to map a group of hosts to some well defined roles, represented by things ansible calls tasks.
- At a basic level, a task is nothing more than a call to an ansible module (see Working With Modules).
- By composing a playbook of multiple ‘plays’, it is possible to orchestrate multi-machine deployments and running certain steps on all machines.

Ansible Variables

- While automation exists to make it easier to make things repeatable, all of your systems are likely not exactly alike.
- On some systems you may want to set some behavior or configuration that is slightly different from others.
- Also, some of the observed behavior or state of remote systems might need to influence how you configure those systems.
- Variables in Ansible are how we deal with differences between systems. To understand variables you'll also want to dig into Conditionals and Loops.

Ansible Variables Definition

Host Variables
[atlanta]
host1 http_port=80 maxRequestsPerChild=808
host2 http_port=303 maxRequestsPerChild=909

Group Variables
[atlanta]
Host1
Host2
[atlanta:vars]
ntp_server=ntp.atlanta.example.com
proxy=proxy.atlanta.example.com

Ansible Loops

Often you'll want to do many things in one task, such as create a lot of users, install a lot of packages, or repeat a polling step until a certain result is reached.
So Instead of doing this:

```
- name: add user testuser1
  user:
    name: "testuser1"
    state: present
    groups: "wheel"
- name: add user testuser2
  user:
    name: "testuser2"
    state: present
    groups: "wheel"
```

You can do this:

```
- name: add several users
  user:
    name: "{{ item }}"
    state: present
    groups: "wheel"
  loop:
     - testuser1
     - testuser2
```

Ansible Conditionals

Sometimes you will want to skip a particular step on a particular host. This could be something as simple as not installing a certain package if the operating system is a particular version, or it could be something like performing some cleanup steps if a filesystem is getting full.

Ansible Roles

Roles are ways of automatically loading certain vars_files, tasks, and handlers based on a known file structure. Grouping content by roles also allows easy sharing of roles with other users.

```
site.yml
webservers.yml
fooservers.yml
roles/
   common/
     tasks/
     handlers/
     files/
     templates/
     vars/
     defaults/
     meta/
   webservers/
     tasks/
     defaults/
     meta/
```

Roles expect files to be in certain directory names. Roles must include at least one of these directories, however it is perfectly fine to exclude any which are not being used. When in use, each directory must contain a main.yml file, which contains the relevant content:

- tasks - contains the main list of tasks to be executed by the role.
- handlers - contains handlers, which may be used by this role or even anywhere outside this role.
- defaults - default variables for the role (see Using Variables for more information).

- vars - other variables for the role (see Using Variables for more information).
- files - contains files which can be deployed via this role.
- templates - contains templates which can be deployed via this role.

- meta - defines some meta data for this role. See below for more details.

Ansible Templates

- Ansible templates used to template a file with different jinja2 options
- For example you can use templates to template a file with different variables value.
- You can use jinja2 conditionals and loops to template a file according to your needs.

```
hello_world.j2
--------------
{{ variable_to_be_replaced }}
This line won't be changed
Variable given as inline - {{ inline_variable }} - :)
```

local_action, Vault, Serial

Sometimes you will have that a specific task will be executed locally on the ansible host running the playbook, for this you can use local_action

```
---
# ...
  tasks:

  - name: Send summary mail
    local_action:
      module: mail
      subject: "Summary Mail"
      to: "{{ mail_recipient }}"
      body: "{{ mail_body }}"
    run_once: True
```

Rolling Update Batch Size

By default, Ansible will try to manage all of the machines referenced in a play in parallel. For a rolling update use case, you can define how many hosts Ansible should manage at a single time by using the serial keyword:

```
- name: test play
  hosts: webservers
  serial: 2
  gather_facts: False
  tasks:
  - name: task one
    comand: hostname
  - name: task two
    command: hostname
```