

DiveInto.com

Presents



Dive Into Ansible

Instructor: James Spurin

Learn and Master Ansible to expedite Scalable
Orchestration, DevOps, and Automation, with the
Complete Ansible Curriculum

Your Instructor

James Spurin

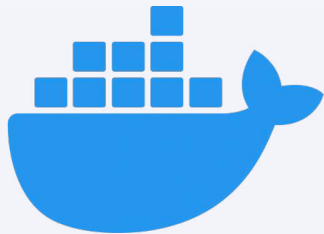


- 20+ Years Industry Experience
- Worked at Nomura, Goldman Sachs, Dell EMC, Hitachi, Virgin Internet
- MSc with Honours in Software Engineering from Liverpool University
- CNCF Kubernetes CKA and CKAD
- Creator of XMLdataset Python Library, featured on the Awesome Python List
- Maintainer of a popular Hexo Docker container image
- Author of Mastering Ansible
- Founder of DiveInto

 linkedin.com/in/jamesspurin



GitHub



docker

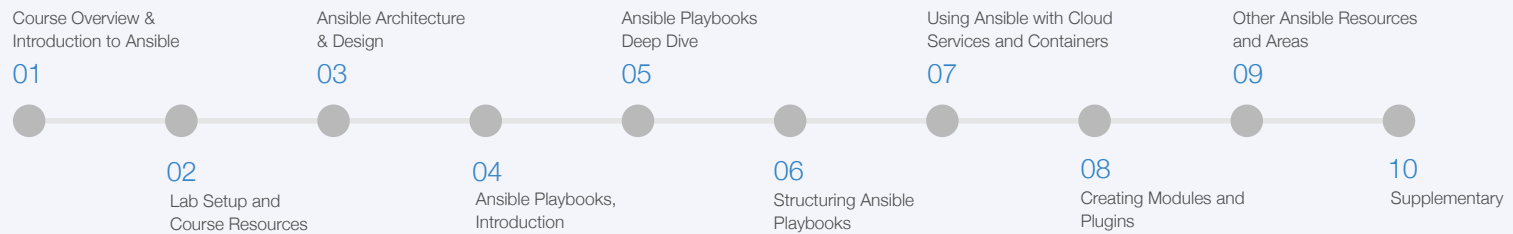


ANSIBLE

Course Resources

- GitHub Repository, 200+ Ansible Examples
- Docker Based Ansible Lab
 - Dedicated Ansible Host
 - 3 x Ubuntu Hosts
 - 3 x CentOS Hosts
 - Reverse Web Proxy
 - Web Terminal & SSH Access

Course Content



=

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Introduction to Ansible

Ansible History

Developed by
Michael DeHaan
2012

Acquired by
Red Hat, Inc
2015

Open Source
Community Driven
Today



Ansible Core Components

Modules



https://docs.ansible.com/ansible/2.9/modules/list_of_all_modules.html



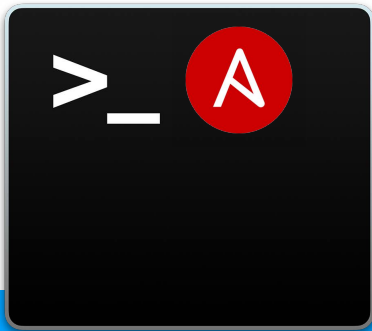
- Cloud Computing
- Networking
- Server Configuration and Management
- Virtualisation
- Containers
- Much More



We can create our own modules, using the Ansible Framework. Covered in this course!

Ansible Core Components

Ansible Executable



Typing `ansible`, at a prompt where Ansible is installed



Great for initially setting up a project and testing our Ansible configuration



Easy to use, with Ansible Modules

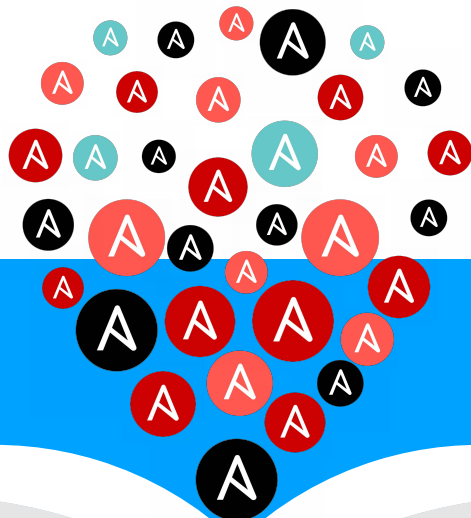


Easy to use, with your Ansible Target Infrastructure



Ansible Core Components

Ansible Playbook



Typing `ansible-playbook`, at a prompt where Ansible is installed



Use Ansible's Human Readable Configuration
Deployment and Orchestration Language



A book of plays



Ansible Core Components

Ansible Inventories



An inventory of targets



Hosts, Network Switches, Containers,
Storage Arrays and More...



Useful information



Dynamic Inventories



DiveInto.com
Presents



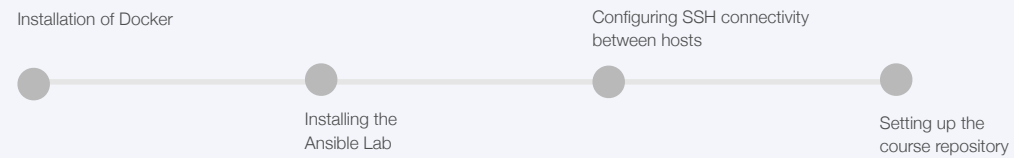
Dive Into Ansible

Instructor: James Spurin

Section 2: Setup of the Lab Environment and Course
Resources

Installing Docker

Section Overview



DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Installing the Ansible Lab

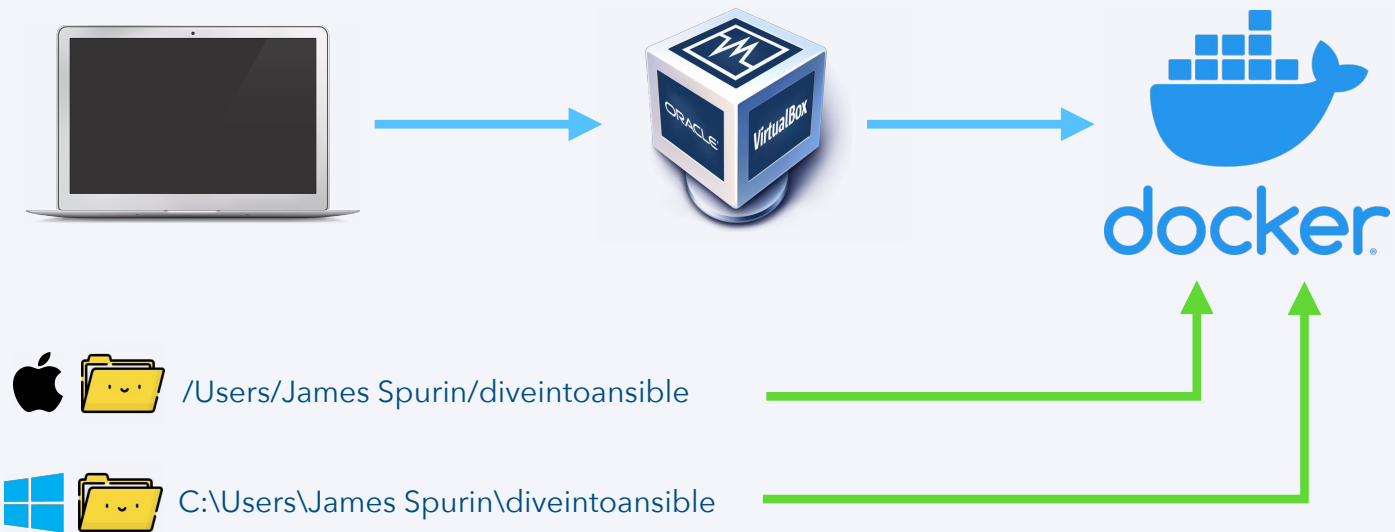


GitHub

Course Lab Repository

- <https://github.com/spurin/diveintoansible-lab>

Docker Desktop Setup



DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Configuring SSH connectivity between hosts

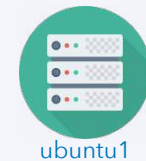
SSH Connectivity Overview



```
ubuntu-c login: ansible
Password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 4.19.76-linuxkit x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

ansible@ubuntu-c:~$ ssh ubuntu1
The authenticity of host 'ubuntu1 (172.18.0.7)' can't be established.
ECDSA key fingerprint is SHA256:07+JTsw7ZHG+81o0bQG4U+2Nu8XLhSzcIR2GbqZC0cQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? █
```



SSH Connectivity Overview



1. ubuntu-c (client) requests an SSH session with ubuntu1 (server)

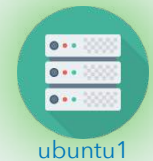
2. SSH Protocol versions are exchanged, if they are compatible they agree, otherwise, no connection

3. Key Exchange, information is shared with the cryptographic primitives that each side supports

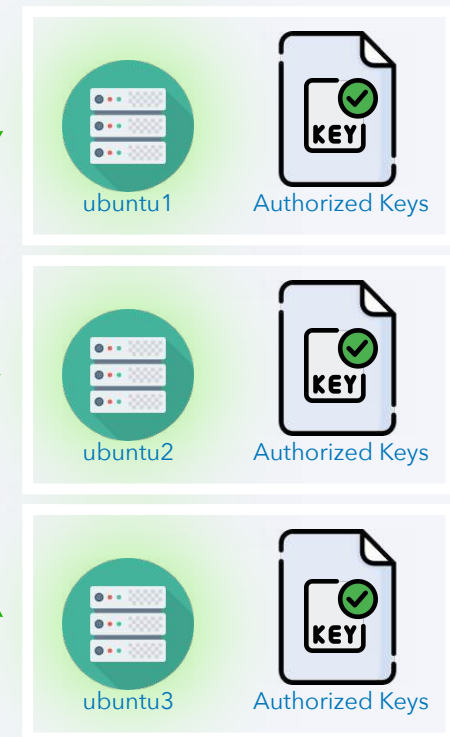
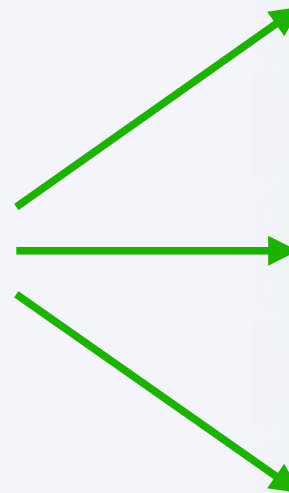
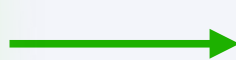
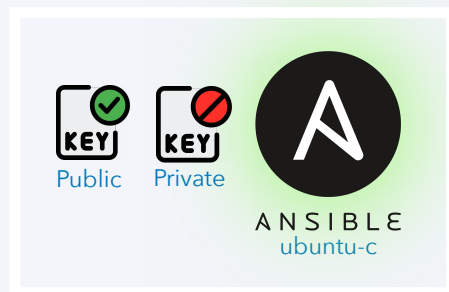
4. Both Client and Server, negotiate session key, using Diffie-Helman algorithm, creating a symmetric key. At this point, if the server is not known to the client, you will see the request to add the key fingerprint to the clients known hosts file

5. New keys between client & server, future communication uses these keys

6. Encrypted session established, ready for authentication



Ansible Connectivity



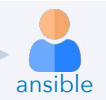
Passwordless Connectivity



ansible



ubuntu1



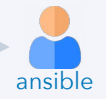
ansible



root



ubuntu2



ansible



root



ubuntu3



ansible



root



centos1



ansible



root



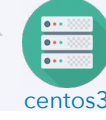
centos2



ansible



root



centos3



ansible



root

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Setting up the Course Repository

Setting up the course repository



- Course Code Repository - <https://github.com/spurin/diveintoansible>
- Setup on ubuntu-c - `git clone https://github.com/spurin/diveintoansible.git`

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Section 1 & 2 Quiz

Let's check our Ansible Knowledge



Who Invented Ansible?

Answer: **Michael DeHaan**

What year, was Ansible created in?

Answer: **2012**



When, was Ansible Acquired, by Red Hat, Inc

Answer: **2015**



Ansible is a single tool, True or False

Answer: **False** - **Ansible, is a toolset, comprising of many tools, modules and is also an extensible framework**



Name some of the core components, of Ansible

Answer:

Modules

ansible executable

ansible-playbook executable

Inventories

... many more

Let's check our Ansible Knowledge



What are the types of targets, we can use with Ansible?

Answer:

Hosts

Network Switches

Containers

Storage Arrays

... many more



When SSH is used, during the secure channel configuration, what algorithm is used to create a symmetric key

Answer: **Diffie-Helman**



What command is used, to generate a public and private ssh keypair

Answer: **ssh-keygen**



In what file, should a public key be added to on a remote users home directory, to configure trusted access:

Answer: **authorized_keys**

Let's check our Ansible Knowledge



In which directory, within a users home directory, would you find a generated public and private ssh key, a known_hosts file and the authorized_keys file
Answer: **.ssh**



What is the name of the convenient ssh tool, we can use for copying our public key, to a target users authorized_keys file
Answer: **ssh-copy-id**



What is the SSH option, for automatically accepting, unknown Host Key Fingerprints
Answer: **StrictHostKeyChecking=no**



On the command, "ansible -i,ubuntu1 -m ping" what does the -i and -m option represent
Answer: **-i is used for the inventory, -m is used for the module**

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

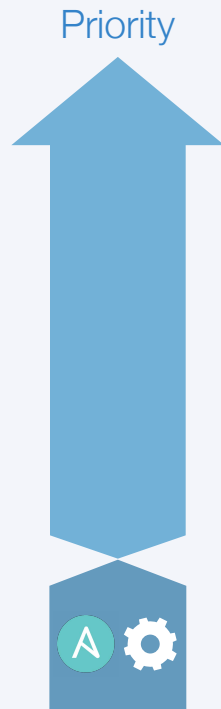
Section 3: Ansible Architecture and Design

Ansible Configuration

Section Overview



Ansible Configuration Files



1. `ANSIBLE_CONFIG` (Environment Variable, with a filename target)
2. `./ansible.cfg` (An `ansible.cfg` file, in the current directory)
3. `~/.ansible.cfg` (A hidden file, called `.ansible.cfg`, in the users home directory)
4. `/etc/ansible/ansible.cfg` (Typically provided, through packaged or system installations of Ansible)

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

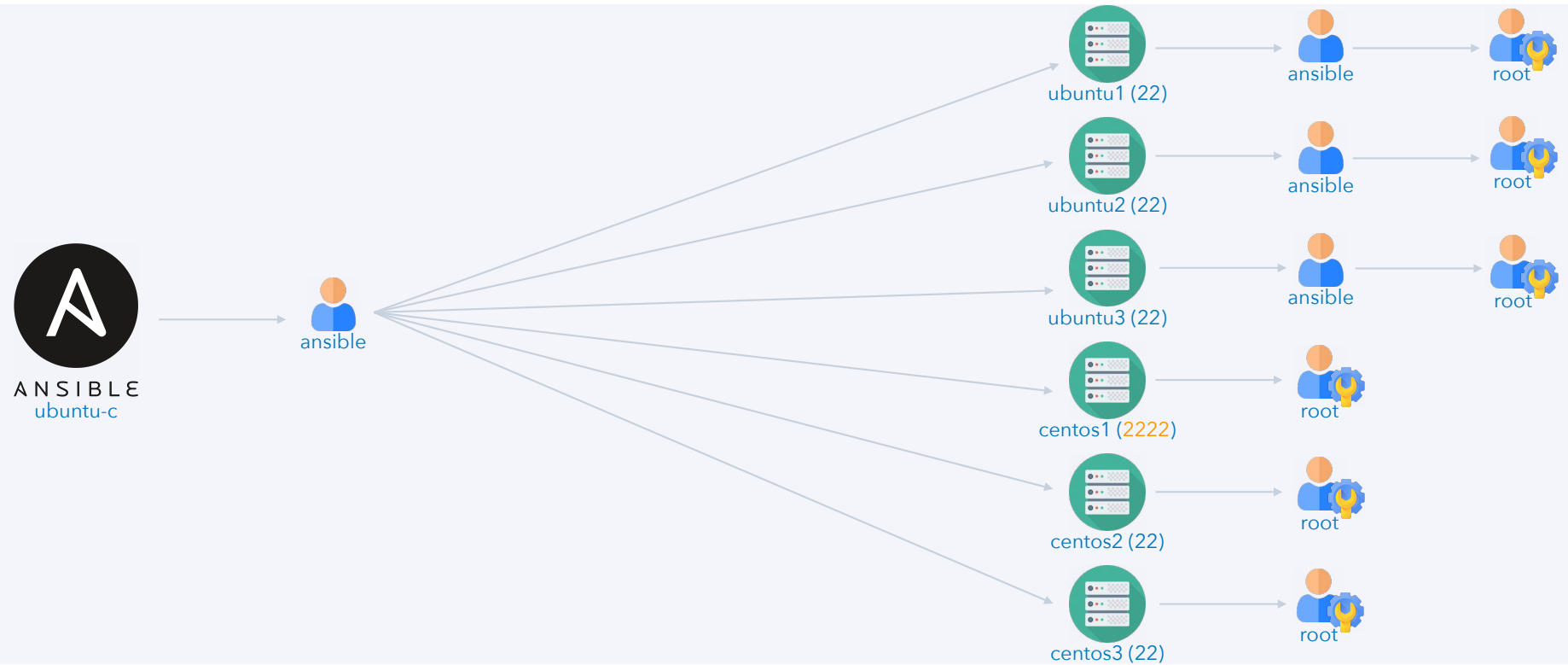
Ansible Inventories

Ansible Inventories

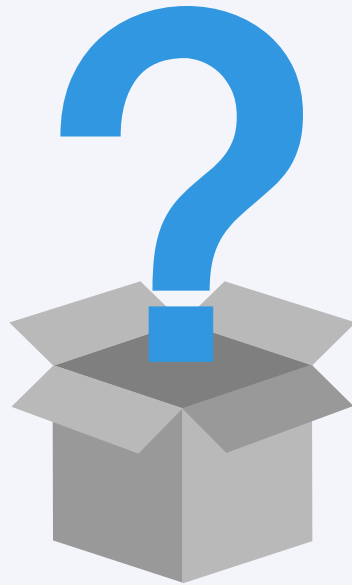


- Ansible Inventories
- Provide Ansible connectivity to our centos hosts via root
- Provide Ansible connectivity to our ubuntu hosts via sudo
- Inventory host variables (hostvars)
- Simplification of Inventory with ranges
- Inventory group variables (groupvars)
- Inventory children groups

Ansible Inventory



Let's check our Ansible Knowledge



In the challenge directory -

Create your own `ansible.cfg` configuration file, it should have a `defaults` section, specify the path to the inventory and `host_key_checking`, should be false



The inventory file, should be called `myinventory`



Create a group, for hostnames ending in even numbers, and a group for hostnames ending in odd numbers



* Remember, that `centos1` uses port `2222`, use the `ansible_port` hostvar, for this host

Create a parent group, called `oddsandevens` that inherits, both the even group, and the odd group

Lastly, make the `ubuntu` hosts, use the `root` account directly and make the `centos` hosts, become `root`, via the `ansible` user

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Ansible Modules

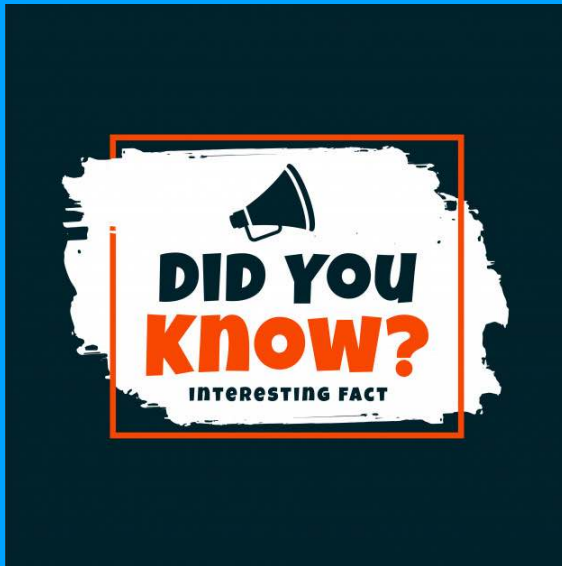
Ansible Modules



- Ansible Modules
- The setup module
- The file module
- Color notation used during Ansible execution
- Idempotence
- The copy module
- The command module
- Ansible-doc

Setup Module

Used for gathering facts when
executing playbooks



- This module is automatically executed, when using playbooks to gather useful information as variables, about remote targets. The information, can be used, during execution
- The module, can also be executed directly, by the ansible command to find out the variables, available to a host
- Ansible provides many 'facts' about a target automatically
- This module, is also supported for Windows targets
- In Ansible 2.10, this has been moved to ansible-base and is classed as a 'builtin' plugin. It can be referenced via the name 'setup' or 'ansible.builtin.setup'
- Documentation - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/setup_module.html

File Module

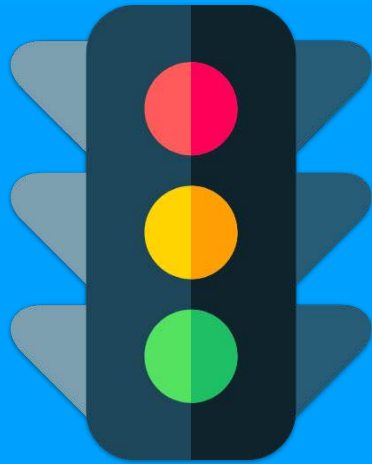
Used for file, symlinks and directory manipulation



- Sets attributes of files, symlinks and directories, or, removes files, symlinks and directories
- Many other modules support the same options as the 'file' module, including [copy], [template] and [assemble]
- For Windows targets, use the [win_file] module instead
- In Ansible 2.10, this has been moved to ansible-base and is classed as a 'builtin' plugin. It can be referenced via the name 'file' or 'ansible.builtin.file'
- Documentation - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/file_module.html

Ansible Colors

Signifies Success or Failure, with or without changes



- Red = Failure
- Yellow = Success, with Changes
- Green = Success, no Changes

Unix Permissions

User / Group / Other
RWX / 421



User

Group

Other

RWX

421

RWX

421

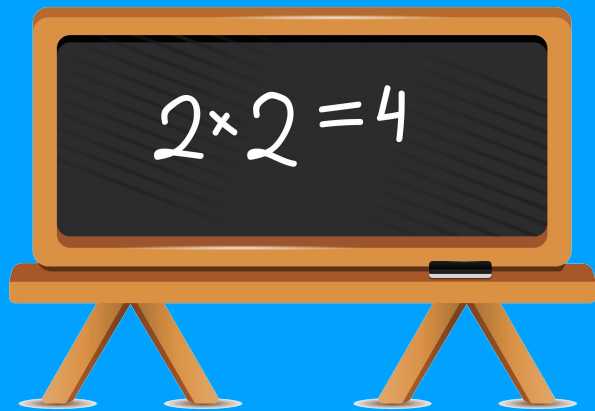
RWX

421

Permission 600 =

RW-

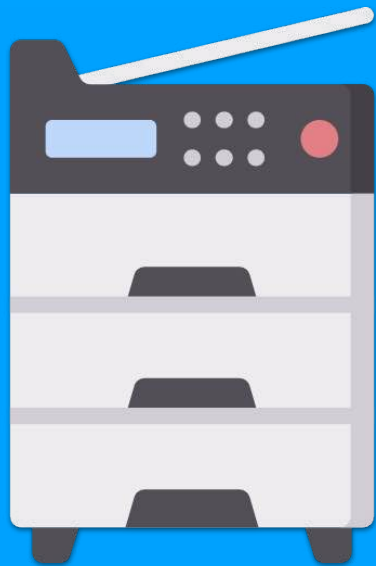
Idempotency



An operation is idempotent, if the result of performing it once, is exactly the same as the result of performing it repeatedly without any intervening actions.

Copy Module

Used for copying files, from the local or remote, to a location on the remote



- The 'copy' module copies a file from the local or remote target, to a location on the remote target. Use the [fetch] module, to copy files from a remote target, to a local target
- If you need variable interpolation in the copied files, use the [template] module.
- For Windows targets, use the [win_copy] module instead
- In Ansible 2.10, this has been moved to ansible-base and is classed as a 'builtin' plugin. It can be referenced via the name 'copy' or 'ansible.builtin.copy'
- Documentation - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/copy_module.html

Command Module

Used for executing remote commands



- The 'command' module, takes the command name followed by a list of space-delimited arguments.
- The given command will be executed on all selected nodes
- It is not processed through the shell, so, variables like \$HOME and operations like <, >, |, ; and &, will not work. Use the [shell] module if you need these features
- For Windows targets, use the [win_command] module instead
- In Ansible 2.10, this has been moved to ansible-base and is classed as a 'builtin' plugin. It can be referenced via the name 'command' or 'ansible.builtin.command'
- Documentation - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/command_module.html

Let's check our Ansible Knowledge



Take a look at

https://docs.ansible.com/ansible/latest/collections/ansible/builtin/fetch_module.html



Firstly, using the file module, create a file on all remote hosts, called `/tmp/test_modules.txt` with a permission of 600



Then, use the fetch module to copy the file from the remote system, to the local system



ansible-doc

Manual Pages for Ansible



- Convenient and great alternative, to the online docs

- Try -

`ansible-doc file`

`ansible-doc fetch`

- Also provides you with the source code location, for modules, allowing you to understand, how a module works behind the scenes

DiveInto.com
Presents



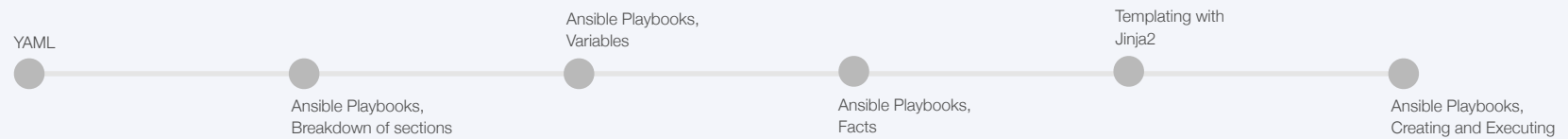
Dive Into Ansible

Instructor: James Spurin

Section 4: Ansible Playbooks, Introduction

YAML

Section Overview



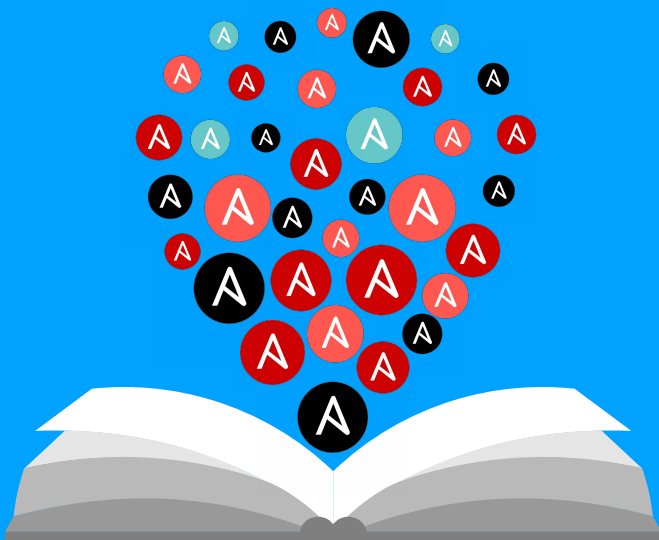
YAML



- YAML is a data-oriented language
- Structure of YAML Files
- Indentation
- Quotes, advantages and disadvantages
- Multiline values
- Boolean (True or False)
- Lists and Dictionaries

Playbooks

YAML ...
"YAML Ain't Markup Language"



- Ansible Playbooks, utilise YAML, as a human readable, data-serialisation language
- Easy to use, easy to read, great for collaboration
- Reading and Writing of YAML, supported in most major programming languages
- Often seen with a .yaml or .yml extension, .yaml is officially the recommended extension since 2006

Let's check our Ansible
Knowledge



YAML Challenge

1. Create a file called test.yaml, with the appropriate start and end markers, run the test utility, the output, should be None



2. Create a list of Car Manufacturers, it should include:



Aston Martin
Fiat
Ford
Vauxhall



3. Change each of these entries in the list, so that they are dictionaries

4. Add as key values to each manufacturer, year_founded and website

5. Add a key of founded_by, but with a list of the founder, or founders

Resources

Useful References



- YAML Specification - <http://yaml.org/spec/1.2/spec.html>
- Wikipedia - <https://en.wikipedia.org/wiki/YAML>
- Other options for working with multi line input/output in YAML -

<https://stackoverflow.com/questions/3790454/in-yaml-how-do-i-break-a-string-over-multiple-lines>

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Ansible Playbooks, Breakdown of Sections

Ansible Playbooks, Breakdown of Sections



- Understand Common Sections
- Create a 'Message of the Day' Playbook
- Use variables
- Override Variables, from the command line
- Use Handlers, for post task execution
- Update Playbook, to Target both CentOS and Ubuntu, with the when directive

Let's check our Ansible Knowledge



Playbooks Challenge

1. Change to the challenge directory, and either create, or, copy the `ansible.cfg` and `hosts` file. Check that our hosts are reachable using the `ansible` command



2. Copy the `motd_playbook.yaml` template from revision 01



3. Update the playbook, to target the `ubuntu` group

4. Add a Handler that will debug, if there is a change



5. Create a task, that copies the file in the currently directory, `60-ansible-motd`, to `/etc/update-motd.d/60-ansible-motd`

Use the option `mode`, for `copy`, to set permissions to `preserve`

! Remember, to add a `notify` option to the task to inform of changes

Resources

Useful References



- Playbooks Keywords - http://docs.ansible.com/ansible/devel/playbooks_keywords.html

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Ansible Playbooks, Variables

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Ansible Playbooks, Facts

Ansible Playbooks, Facts



- The setup module, and how this relates to fact gathering
- Filtering for specific facts
- The creation of custom facts
- The execution of custom facts
- How custom facts can be used in environments, without super user access

Resources

Useful References



- Setup Module - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/setup_module.html

Custom Facts



- Can be written in any language
- Returns a JSON structure
- (or) Returns an ini structure
- By default, expects to use `/etc/ansible/facts.d`

Custom Facts



- JSON Output -

```
{"date" : "Day Mon DD HH:MM:SS TZ YYYY"}
```

- INI Output -

```
[date]
```

```
date=Day Mon DD HH:MM:SS TZ YYYY
```


DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Templating with Jinja2

Templating with Jinja2



- The Jinja2 Templating Language
- If / elif / else statements
- for loops
- break and continue
- ranges
- Jinja2 filters

Resources

Useful References



- Filters - https://docs.ansible.com/ansible/latest/user_guide/playbooks_filters.html

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Ansible Playbooks, Creating and Executing

Ansible Playbooks, Creating and Executing



- A hands on, use case example where we create and executing Ansible Playbooks, through an exciting and interesting project
- Install Nginx webserver on both CentOS and Ubuntu and we'll explore the different options for targeting OS variations with Ansible Packages
- Use Ansible, to install and configure our project website, taking into account variances between the Linux distributions of Nginx on both CentOS and Ubuntu
- Use Jinja2 templates, to customise our website
- We'll explore the use of the 'Ansible Managed' functionality
- We'll install a secret "Easter Egg" into our website application

Let's check our Ansible Knowledge



Playbooks Creating and Executing Challenge

1. Configure our hosts to target the linux group
2. CentOS/RHEL uses yum or it's successor dnf, for package installation. Install the package named epel-release using either the yum or dnf module

Give the task a name of 'Install Epel'

Use the following options -

update_cache: yes
state: latest

Target only the CentOS hosts for this using the fact ansible_distribution with a value of CentOS



Let's check our Ansible Knowledge



Playbooks Creating and Executing Challenge

3. Create a task called 'Install Nginx CentOS'

Use the yum or dnf module, to install a package named nginx

Use the same update_cache and state options as before

Use a fact, to check the ansible_distribution is CentOS

4. Create a task called 'Install Nginx Ubuntu'

Use the apt module, to install a package named nginx

Use the same update_cache and state options as before

Use a fact, to check the ansible_distribution is Ubuntu



Let's check our Ansible Knowledge



Playbooks Creating and Executing Challenge

5. Remove the existing Nginx tasks and create a task called 'Install Nginx'

Use the package module, to install a package named nginx

Use the option of state: latest

6. Create a task called 'Restart nginx'

Use the service module to target nginx, with a state of 'restarted'



Let's check our Ansible Knowledge



Playbooks Creating and Executing Challenge

7. Create a handler named 'Check HTTP Service'

Use the uri module, with the following parameters

url: `http://{{ ansible_default_ipv4.address }}`

status_code: 200

8. Update the 'Restart nginx' task, to notify 'Check HTTP Service'



Let's check our Ansible Knowledge



Playbooks Creating and Executing Challenge

11. Update the ansible.cfg file and include the following

```
ansible_managed = Managed by Ansible - file:
{file} - host:{host} - uid:{uid}
```

12. Update the templating task to 'Template index.html-ansible_managed.j2 to index.html on target'

And update the template file to

```
src: templates/index.html-ansible_managed.j2
```



Let's check our Ansible Knowledge



Playbooks Creating and Executing Challenge

13. Update the playbook to include the vars file

`vars/logos.yaml`

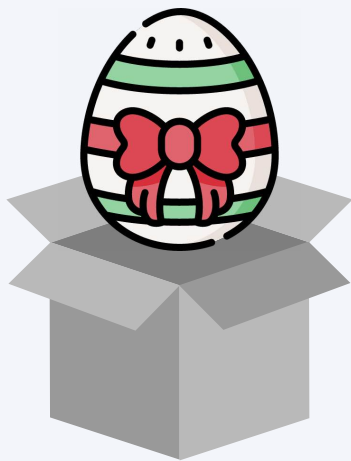
14. Update the templating task to 'Template `index.html-logos.j2` to `index.html` on target'

And update the template file to

`src: templates/index.html-logos.j2`



Let's check our Ansible Knowledge



Playbooks Creating and Executing Challenge

15. Install the unzip package
16. Create a task called 'Unarchive playbook stacker game'



Use the unarchive module, with the following parameters

`src: templates/playbook_stacker.zip`

`dest: "{{ nginx_root_location }}"`

`mode: 0755`



17. Update the templating task to 'Template index.html-easter_egg.j2 to index.html on target'

And update the template file to

`src: templates/index.html-easter_egg.j2`



DiveInto.com
Presents



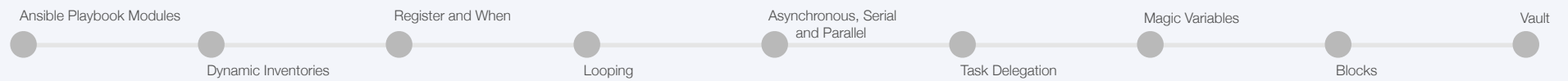
Dive Into Ansible

Instructor: James Spurin

Section 5: Ansible Playbooks, Deep Dive

Ansible Playbook Modules

Section Overview



Playbook Modules



- set_fact
- pause
- prompt
- wait_for
- assemble
- add_host
- group_by
- fetch

Set Fact

Used for gathering facts when
executing playbooks



- Dynamically add or change facts during execution

Pause

*The right word may be effective, but no word was ever as effective as a rightly timed pause - **Mark Twain***



- Pause a playbook for a set amount of time or until a prompt is acknowledged

Assemble

Assemble configuration files from fragments



- Allows configuration files to be broken into segments and concatenated to form a destination file

Great to use, when an application or tool, requires it's configuration as a single file, but, you wish to manage it as separate entities

Add Host

Dynamically add targets to our running playbooks



- Create new inventory groups and targets, on the fly
- Great, for when a resource is created during execution and you wish to include it, in your playbook execution

Group By

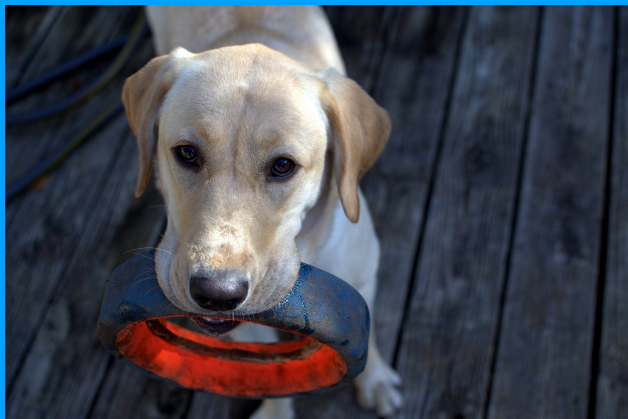
Create groups, based on facts



- Utilise facts to dynamically create, associated groups

Fetch

Capture files



- Capture files from remote hosts and targets

DiveInto.com

Presents



Dive Into Ansible

Instructor: James Spurin

Dynamic Inventories

Video Overview

Dynamic Inventories



- The requirements of Dynamic Inventories
- How to create a Dynamic Inventory with minimal scripting
- How to interrogate a Dynamic Inventory
- Performance enhancements through the use of `_meta`
- The use of the Ansible Python framework for Dynamic Inventories

Recap

Inventories



We've used an inventory of hosts defined via our `ansible.cfg` file



We've associated inventories variables both inline and via the `host_vars` `group_vars` directories



An inventory can be specified or overridden on the command line using `-i`

Dynamic Inventory Key Requirements



- Needs to be an executable file. Can be written in any language providing that it can be executed from the command line
- Accepts the command line options of --list and --host hostname
- Returns a JSON encoded dictionary of inventory content when used with --list
- Returns a basic JSON encoded dictionary structure for --host hostname

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Register and When

Register and When



- How to register output, with the register directive
- How to use registered output
- How to work around differences with registered output
- Filters, that relate to registered content
- Utilising when with register

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Looping

Looping



- `with_items`
- `with_dict`
- `with_subelements`
- `with_together`
- `with_sequence` ... many other loops ... `with_random_choice`
- `until`

DiveInto.com

Presents

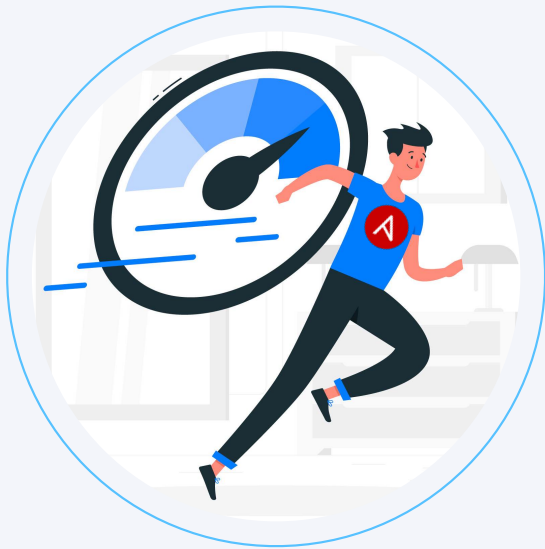


Dive Into Ansible

Instructor: James Spurin

Asynchronous, Serial, Parallel

Asynchronous, Serial, Parallel



- Playbook performance and bottlenecks
- Polling
- Asynchronous job identifiers
- Asynchronous status handling
- Serial execution
- Batch execution
- Alternative strategies to facilitate Parallel execution

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Task Delegation

Task Delegation



- How we can delegate specific tasks, for execution on specific targets
- We'll target our host, Ubuntu3 and through the use of TCP Wrappers, we'll restrict SSH access so that it only works, from ubuntu-c, centos1 and ubuntu1

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Magic Variables

Magic Variables



- Techniques and tricks for accessing and uncovering variables and magic variables, through the use of Ansible playbooks
- Reference: https://docs.ansible.com/ansible/latest/reference_appendices/special_variables.html

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Blocks

Video Overview

Blocks



- How to group multiple tasks, into a single block
- Rescue
- Always

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Vault

Video Overview

Vault



- Encrypting / Decrypting Variables
- Encrypting and Decrypting Files
- Re-Encrypting Data
- Using Multiple Vaults

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Section 6: Structuring Ansible Playbooks

Using Includes & Imports

Section Overview

Using Includes &
Imports



Using Tags

Using Roles



Using Includes & Imports

Using Includes & Imports



- `include_tasks`
- `import_tasks`
- Static vs Dynamic
- `import_playbook`

Static

- Processed as the playbook is parsed
- 'when' statement applies to **all** individual tasks at task point of execution

Dynamic

- Processed at playbook execution
- 'when' statement applies to **all** tasks at initial point of execution

DiveInto.com
Presents

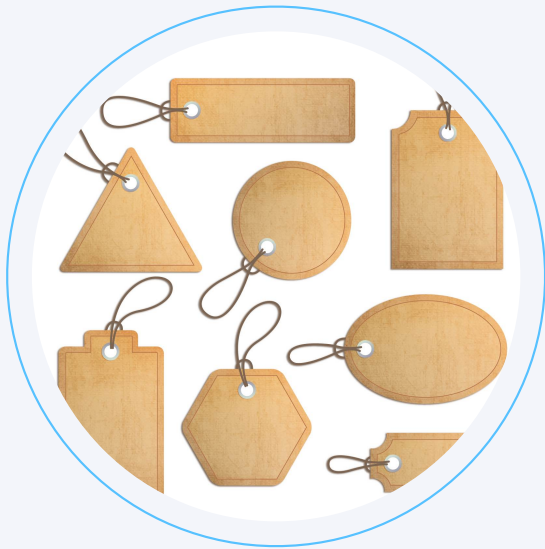


Dive Into Ansible

Instructor: James Spurin

Using Tags

Tags



- Using Tags
- Segmentation with Tags
- Execution with Tags
- Skipping with Tags
- Playbook Tags
- Special Tags
- Tag Inheritance

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Using Roles

Video Overview

Roles



- Using Roles
- The Role Structure
- How to create Roles with Ansible Galaxy
- How to move an existing project, to a Role
- Role Execution
- Role Parameters
- Role Dependencies

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Section 7: Using Ansible with Cloud Services and
Containers

AWS with Ansible

Section Overview

AWS with Ansible



Docker with Ansible

Video Overview

AWS with Ansible



- How to configure Ansible for AWS Support
- Creating instances through Ansible with AWS and adding IP Addresses
- Using the AWS Dynamic Inventory
- Spinning up our WebApp, using AWS
- Terminating and removing AWS Instances

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Docker with Ansible

Docker with Ansible



- Configure our Docker Lab
- Pull Docker Images and Experiment with Docker
- Build Containers
- Build Customised Images and Customised Containers
- Use Ansible to connect to running containers
- Terminate and remove Docker resources

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Section 8: Creating Modules and Plugins

Creating Modules

Section Overview

Creating Modules



Creating Plugins

Video Overview

Creating Modules



- Downloading Ansible Source Code
- Using the Developer 'Hacking' Tools, to Interrogate Modules
- Understand the structure, to report module success and failure
- Build a simple ping module, using shell
- Leverage the Ansible module framework, moving our simple ping module to python
- Show debug information, for failures

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Creating Plugins

Video Overview

Creating Plugins



- Discuss the various types of Plugins
- Create a `lookup_plugin`
- Create a `filter_plugin`

Video Overview

Creating Plugins



- <https://github.com/ansible/ansible/blob/devel/lib/ansible/plugins/lookup/items.py>

Video Overview

Creating Plugins



- https://github.com/ansible/ansible/blob/devel/lib/ansible/plugins/vars/host_group_vars.py

Creating Plugins



Here's a summary of the different types of plugins available, taken from the official ansible documentation:

- Action plugins are front ends to modules and can execute actions on the controller before calling the modules themselves.
- Cache plugins are used to keep a cache of 'facts' to avoid costly fact-gathering operations.
- Callback plugins enable you to hook into Ansible events for display or logging purposes.
- Connection plugins define how to communicate with inventory hosts.
- Filters plugins allow you to manipulate data inside Ansible plays and/or templates. This is a Jinja2 feature; Ansible ships extra filter plugins.
- Lookup plugins are used to pull data from an external source. These are implemented using a custom Jinja2 function.
- Strategy plugins control the flow of a play and execution logic.
- Shell plugins deal with low-level commands and formatting for the different shells Ansible can encounter on remote hosts.
- Test plugins allow you to validate data inside Ansible plays and/or templates. This is a Jinja2 feature; Ansible ships extra test plugins.
- Vars plugins inject additional variable data into Ansible runs that did not come from an inventory, playbook, or the command line.

(http://docs.ansible.com/ansible/latest/dev_guide/developing_plugins.html)

Creating Plugins



On the ansible documentation, it lists the custom directories that we should use for plugins

- action_plugins
- lookup_plugins
- callback_plugins
- connection_plugins
- filter_plugins
- strategy_plugins
- cache_plugins
- test_plugins
- shell_plugins

(http://docs.ansible.com/ansible/latest/dev_guide/developing_plugins.html#distributing-plugins)

Video Overview

Creating Plugins



- https://github.com/ansible/ansible/blob/devel/lib/ansible/plugins/lookup/__init__.py

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Section 9: Other Ansible Resources and Areas

Troubleshooting Ansible

Section Overview

Troubleshooting Ansible



Best Practices with
Ansible

Video Overview

Troubleshooting Ansible



- SSH Connectivity
- Syntax Check
- Step
- Start At
- Log Path
- Verbosity

DiveInto.com
Presents



Dive Into Ansible

Instructor: James Spurin

Best Practices with Ansible

Video Overview

Best Practices with Ansible



- http://docs.ansible.com/ansible/latest/playbooks_best_practices.html