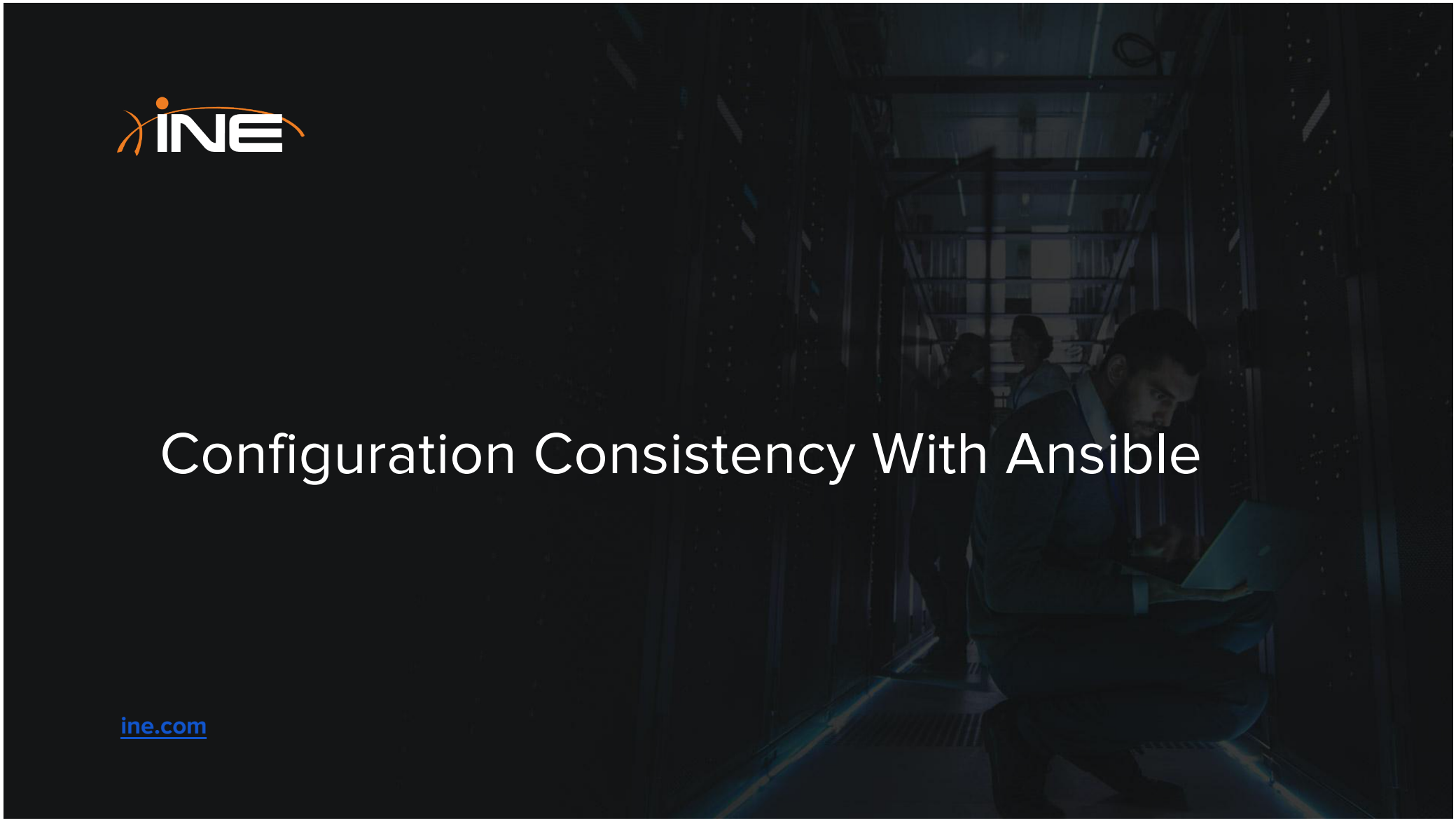


Configuration Consistency With Ansible

ine.com

Keith Bogart

CCIE #4923

kbogart@ine.com

@keithbogart1

linkedin.com/in/keith-bogart-2a75042

CCIE Routing & Switching

Course Objectives

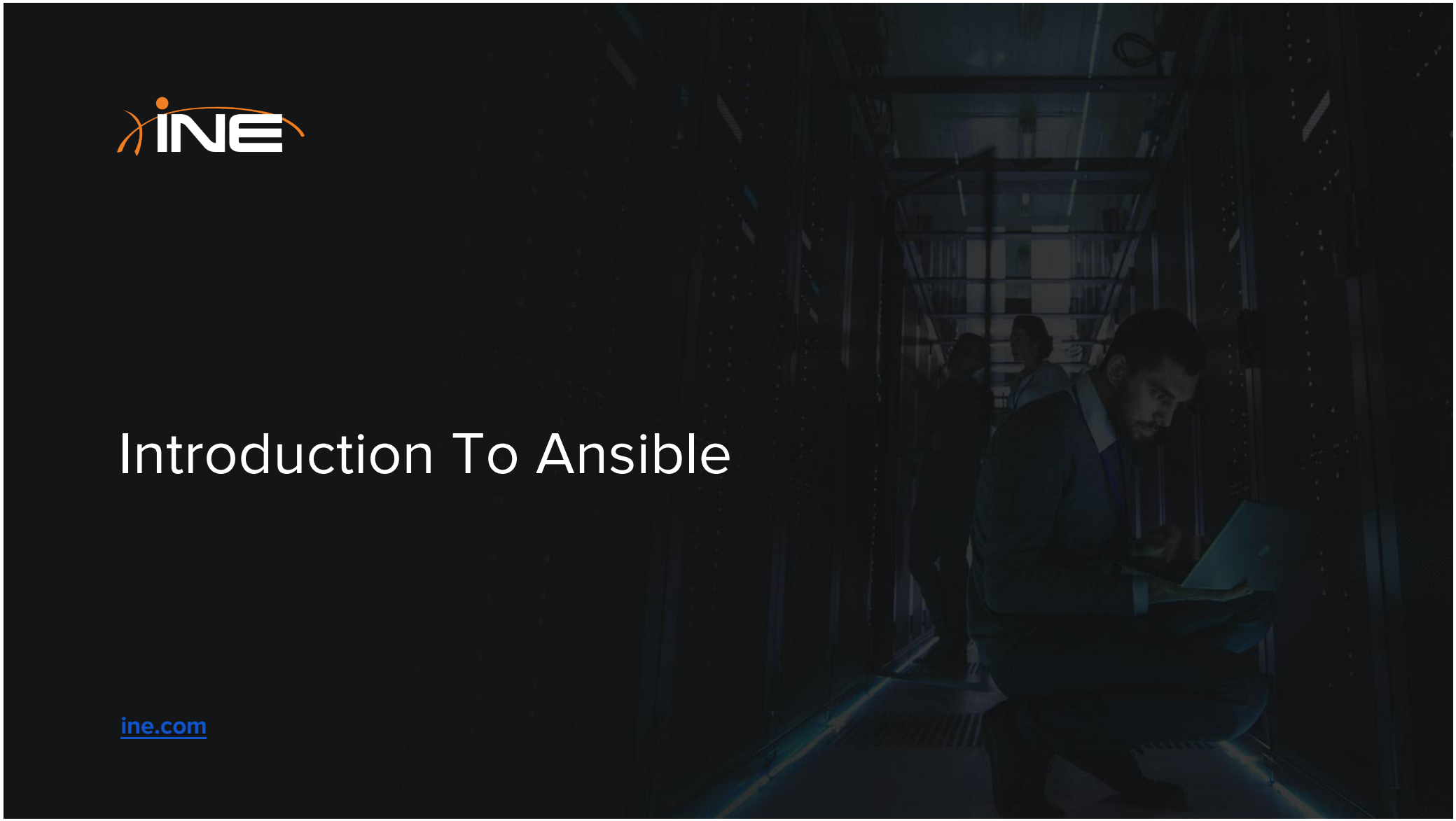
- Introduce You To Ansible Terminology & Concepts
- Provide Detailed Instructions For Downloading & Installing Ansible On Your Own, Local System
- Familiarize You With Basic YAML Syntax Appropriate For Ansible Usage

Course Objectives

+ Provide All Necessary Files & Directions To Lab Infrastructure For Practice
+ Guide You In Creation Of Your First Ansible Playbook
+ Demonstrate Ansible Usage For Saving, Manipulating & Backing-Up Config Files

Course Prerequisites

- The More Experience With Linux...The Better
- Familiarity With Some Network CLI
- Access To Network Devices
- Python Experience Not Necessary, But A Huge Plus!



Introduction To Ansible

ine.com

Topic Overview

+ Overview Of Ansible
+ Ansible For The Network Engineer
+ Overview Of Ansible Tower

What Is Ansible?

- I.T. automation tool
  - Configuration of systems
  - Deployment of software
  - Orchestration of tasks
- Open source
- Originally designed for server management
- A wrapper for Python scripts
- Uses OpenSSH for transport
- Agent-less

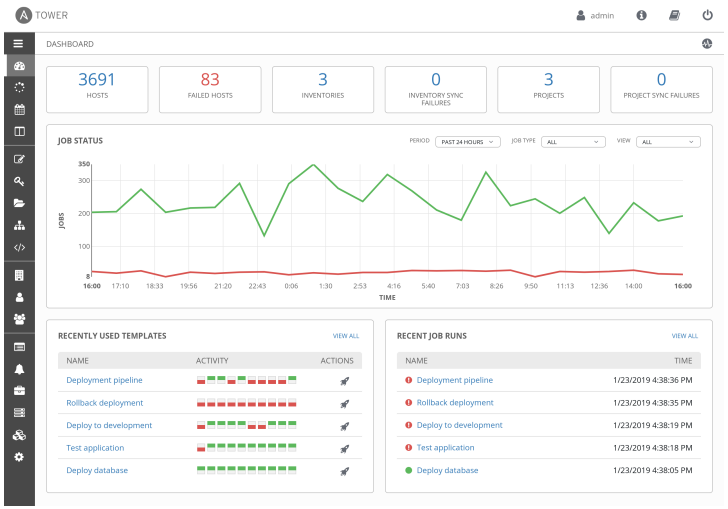
Ansible For The Network Engineer

- Push consistent configurations across multiple devices
- Scan devices for recent configuration changes
- Backup device configurations
- Troubleshoot several devices at once
- Perform multi-device software updates
- ...and more

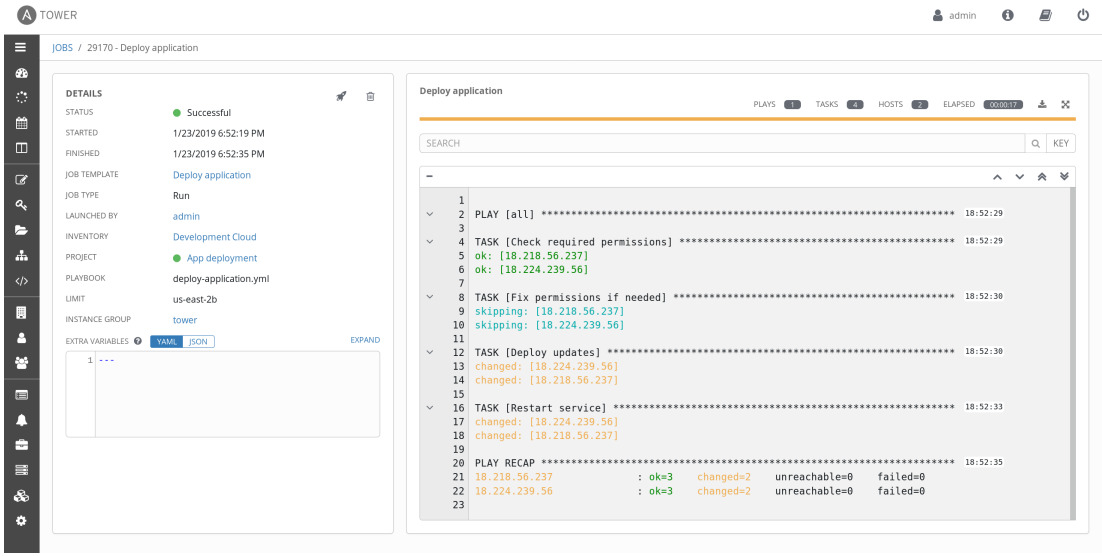
Ansible Tower

- This course is exclusively about the free, open-source version of Ansible utilizing the Linux CLI
- Ansible Tower = paid/licensed version of Ansible providing a GUI
  - Dashboard
  - Multi-Playbook Workflows
  - Logging of Playbook runs
  - Integrated Notifications of job successes or failures
  - Scheduling of Ansible jobs
  - ...and more

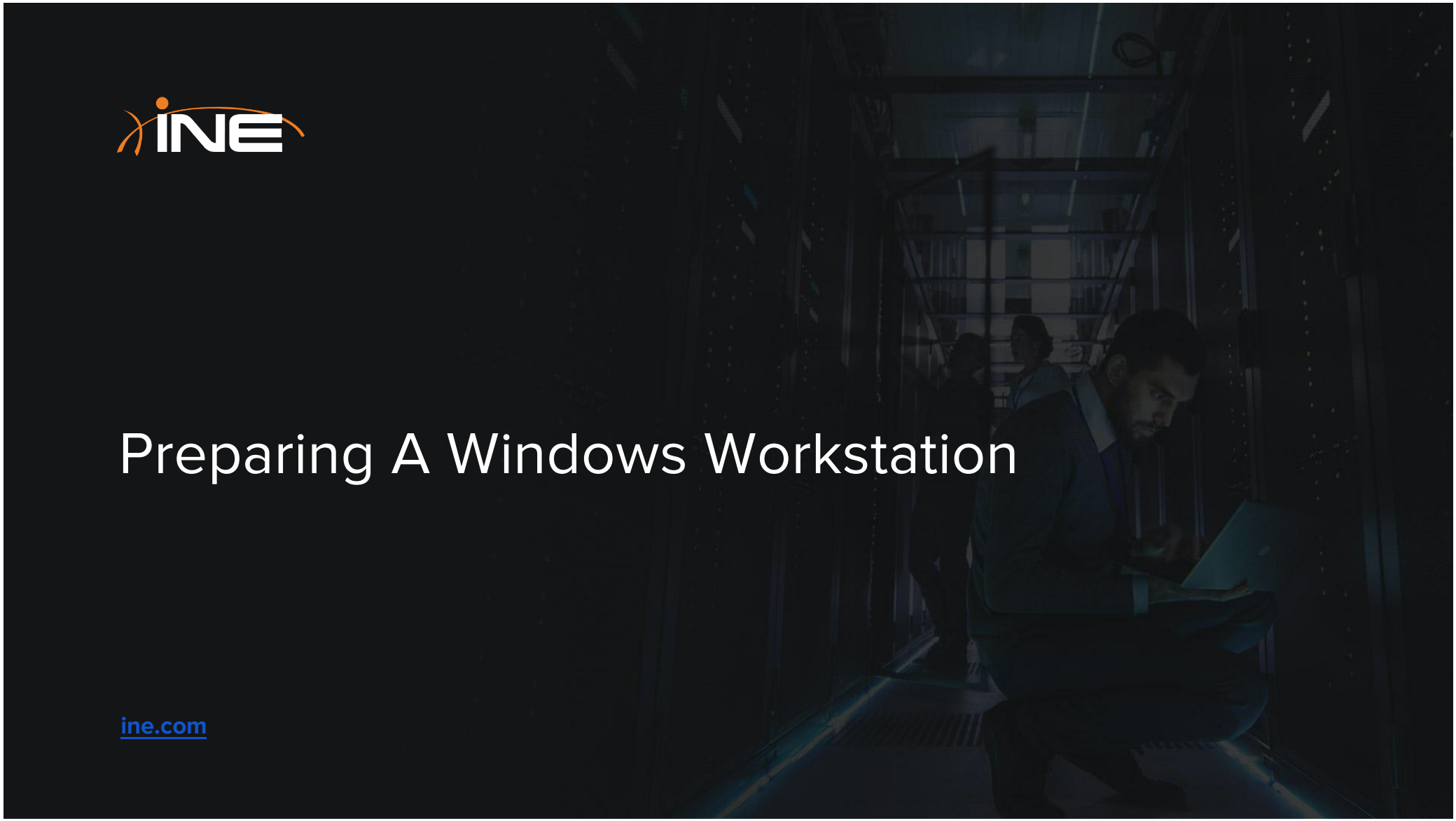
Ansible Tower Resources



https://www.ansible.com/products/tower



Thanks for Watching!



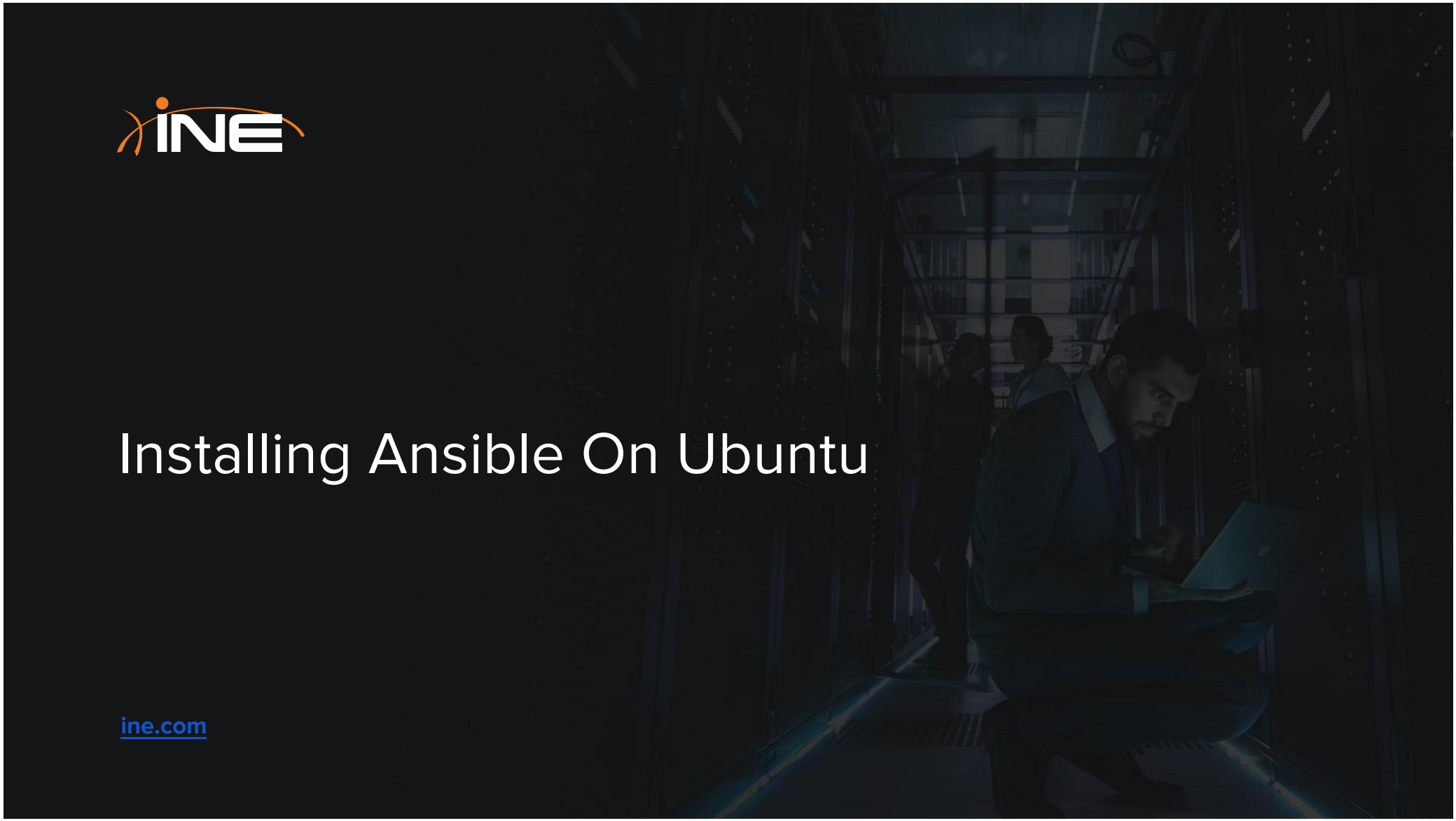
Preparing A Windows Workstation

ine.com

Topic Overview

+ Ansible On Windows – Caveats
+ Virtualizing A Linux Host

Thanks for Watching!



Installing Ansible On Ubuntu

ine.com

Topic Overview

+ Installing Ansible On Ubuntu Control Nodes

Thanks for Watching!

Introduction To Linux CLI Commands

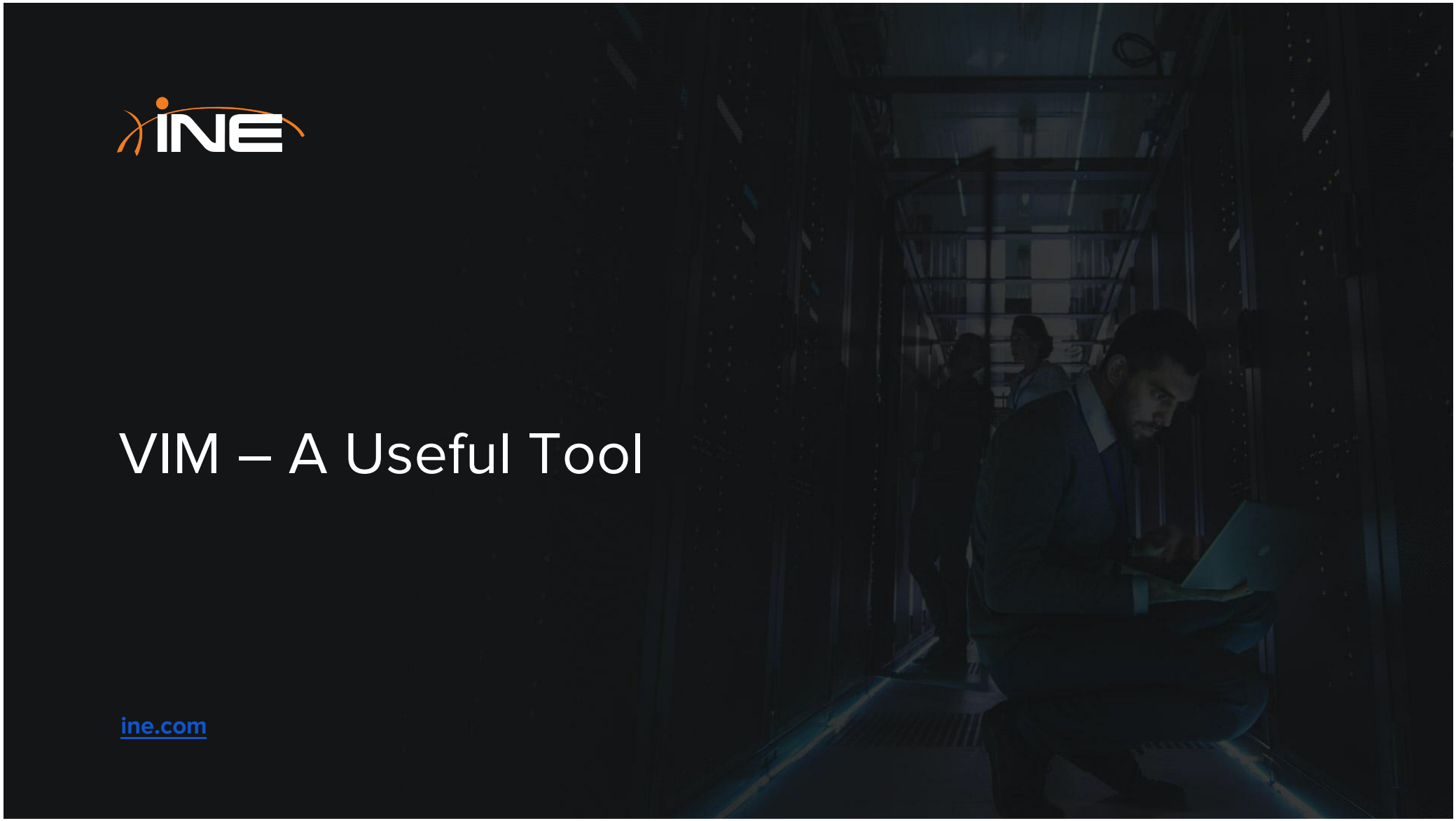
ine.com

Topic Overview

+ Intro To Basic Linux CLI Commands
+ Demonstrations

Commands Covered

- cd and cd ..
- sudo
- sudo mkdir <directory>
- sudo rmdir <directory>
- sudo chmod 777 </path/directory name/>
- cat
- ls
- sudo cp <source> <dest filename>
- sudo mv <source> </path/to/file/new-filename>
- sudo rm ./*



VIM – A Useful Tool

ine.com

Topic Overview

+ Overview Of VIM
+ VIM Installation
+ Useful VIM Shortcuts

What Is VIM

- **Vi Im**proved
  - Vi was an older text editor
- Text editor for Unix/Linux
- Free and Open-Source
- From Wikipedia:
  - “Like vi, Vim's interface is not based on menus or icons but on commands given in a text user interface”

Do You Have VIM?

- You might already have it, to test:
  - Open the Terminal application
  - Type, “apt-cache policy vim”

```
user@user-virtual-machine:/etc/ansible/Devnet-YAML$ apt-cache policy vim
vim:
  Installed: 2:8.0.1453-1ubuntu1.4
  Candidate: 2:8.0.1453-1ubuntu1.4
  Version table:
 *** 2:8.0.1453-1ubuntu1.4 500
        500 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages
        500 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages
        100 /var/lib/dpkg/status
     2:8.0.1453-1ubuntu1 500
        500 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 Packages
user@user-virtual-machine:/etc/ansible/Devnet-YAML$
```

You can also type, “vim --version"

Installing VIM

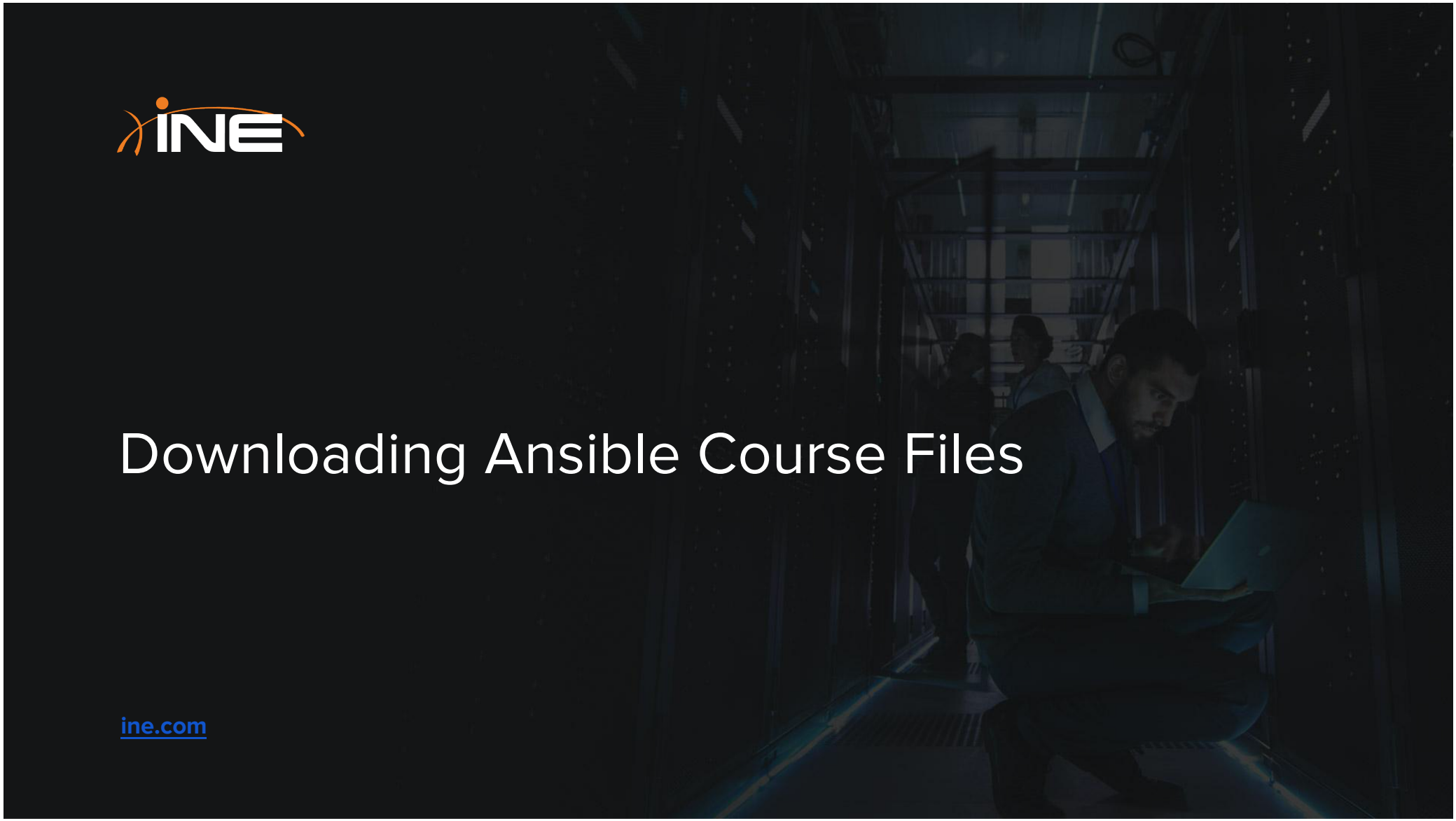
- Run all of the commands below, in order, within the Terminal application:
  - Sudo apt update
    - Alternatively: sudo apt-get update
  - Sudo apt upgrade
    - Alternatively: sudo apt-get upgrade
  - Sudo apt install vim
    - Alternatively: sudo apt-get install vim

Useful VIM Info

- Creating a new file
  - User$ **vim MyTestDocument.txt**
  - **Note: If this document already exists in the current directory, the command above will open it within VIM**
- Exit document without saving changes
  - Press your "**ESC**" key
    - Puts you into "normal mode"
  - Type a colon (:)
  - Type the letter "**q**" followed by the return key

Additional VIM Demonstrations

Thanks for Watching!



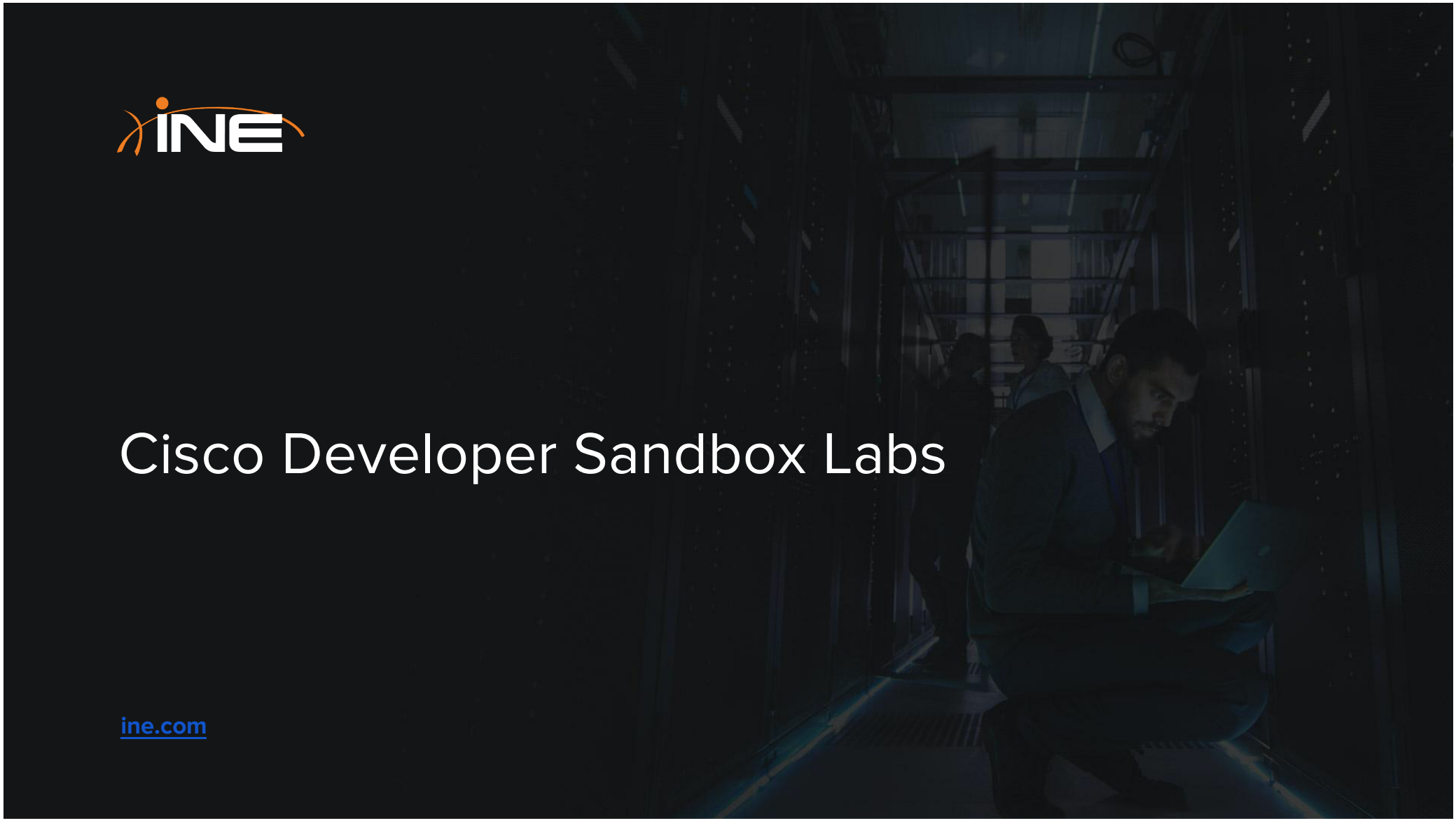
Downloading Ansible Course Files

ine.com

Topic Overview

+ Downloading Ansible Course Files

Thanks for Watching!



Cisco Developer Sandbox Labs

ine.com

Topic Overview

+ Cisco Developer Sandbox
+ Starting A Lab

Locating A Lab

- Some kind of gear is necessary upon which you can test your Ansible commands and Playbooks
- Large topologies are NOT required for learning Ansible
- Cisco Developer Sandbox = FREE LABS!!
- Let’s find and start a lab!

Thanks for Watching!

Obtaining VPN Access To Your Lab

ine.com

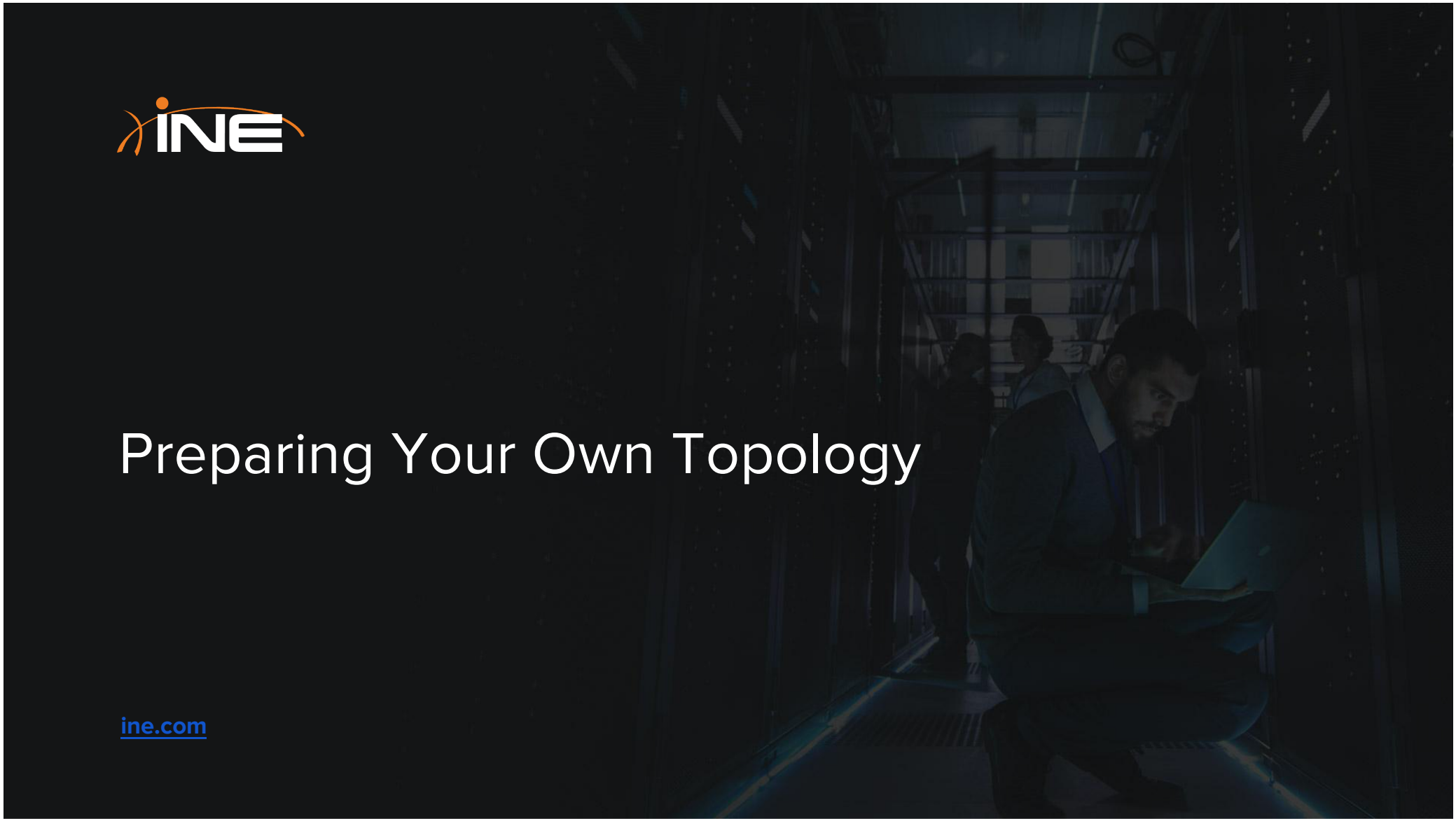
Topic Overview

+ OpenConnect For VPN Access

VPN Access

- Most Cisco Sandbox labs require a VPN client to access the lab
- Cisco recommends Cisco AnyConnect Secure Mobility Client
- OpenConnect also works for this purpose
  - Free
  - Open-Source VPN Client
  - May need to manually disable DTLS

Thanks for Watching!



Preparing Your Own Topology

ine.com

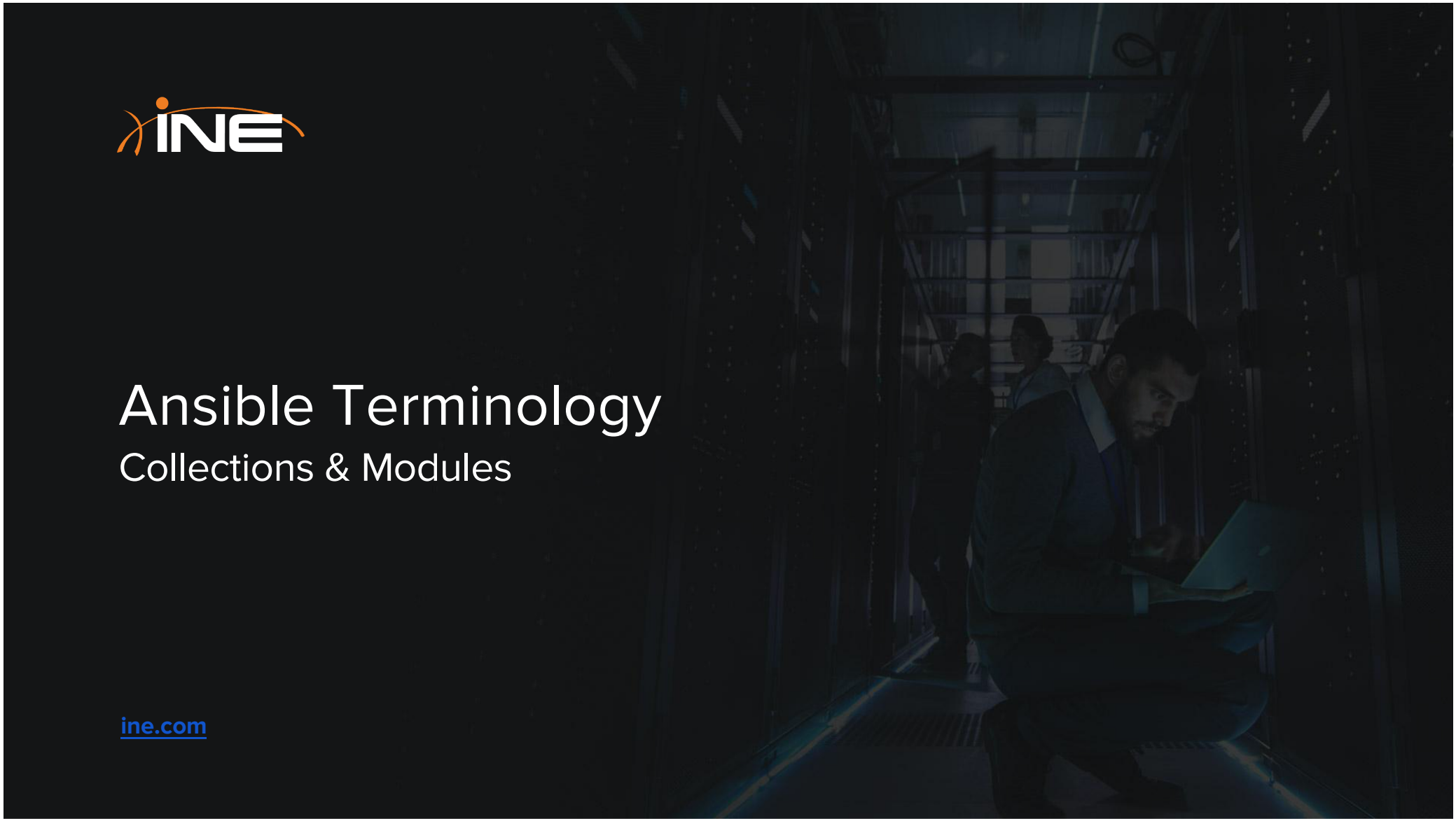
Topic Overview

+ Preparing Your Own Equipment For Ansible

Prepping Your Own Equipment

- Minimum Requirements:
  - Enable SSH access to devices
  - Create SSH credentials
  - Ensure all devices have IP reachability to Ansible Control Node

Thanks for Watching!



Ansible Terminology

Collections & Modules

ine.com

Topic Overview

+ Collections
+ Modules

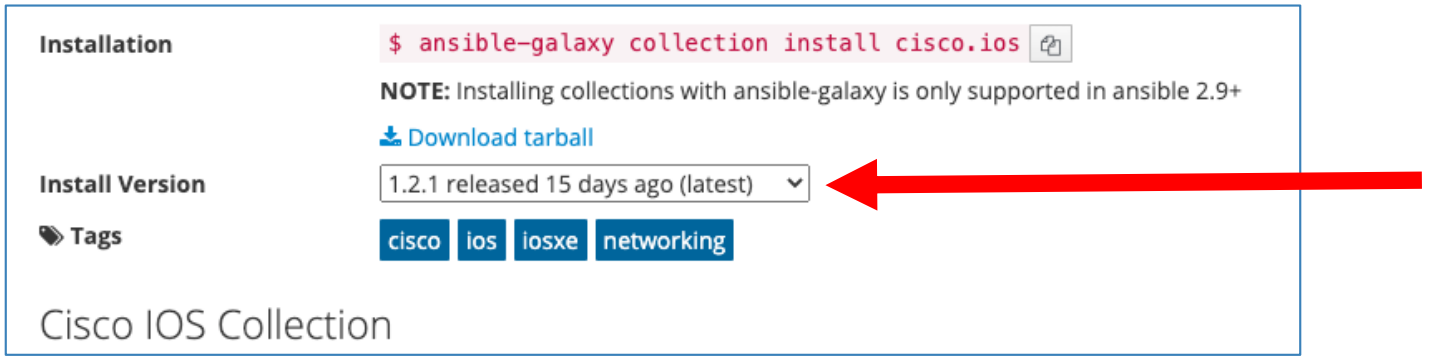
Collections

- A group of modules & plugins, all designed to simplify the automation of devices belonging to a specific vendor
- For the network engineer, some of the most common collections would be:
  - cisco.ios
  - arista.eos
  - f5networks.f5_modules
  - junipernetworks.junos
  - vyos.vyos
  - ...and several others

Locating & Downloading Collections

- Collections are stored at https://galaxy.ansible.com/home
- Search for what you need
- When a collection is identified, download it to your Linux host via:
  - ansible-galaxy collection install <collection name>

Upgrading Installed Collections



- Confirm that you have the most recent collection installed:
  - ansible-galaxy collection install -vv cisco.ios:==1.2.1
- Upgrade to latest version:
  - ansible-galaxy collection install -vv cisco.ios:==1.2.1 --force

Modules

- Discrete units of code (written in Python but referenced via YAML notation) that can be used from the command line or in a playbook task.
- Written to obfuscate the actual commands that are pushed to a device.
  - For example the “ios_acl_interfaces” module contains a variety of parameters (like “name” and “direction”) and values that could be assigned to each parameter (like “No-Payroll” and “in”).

Modules

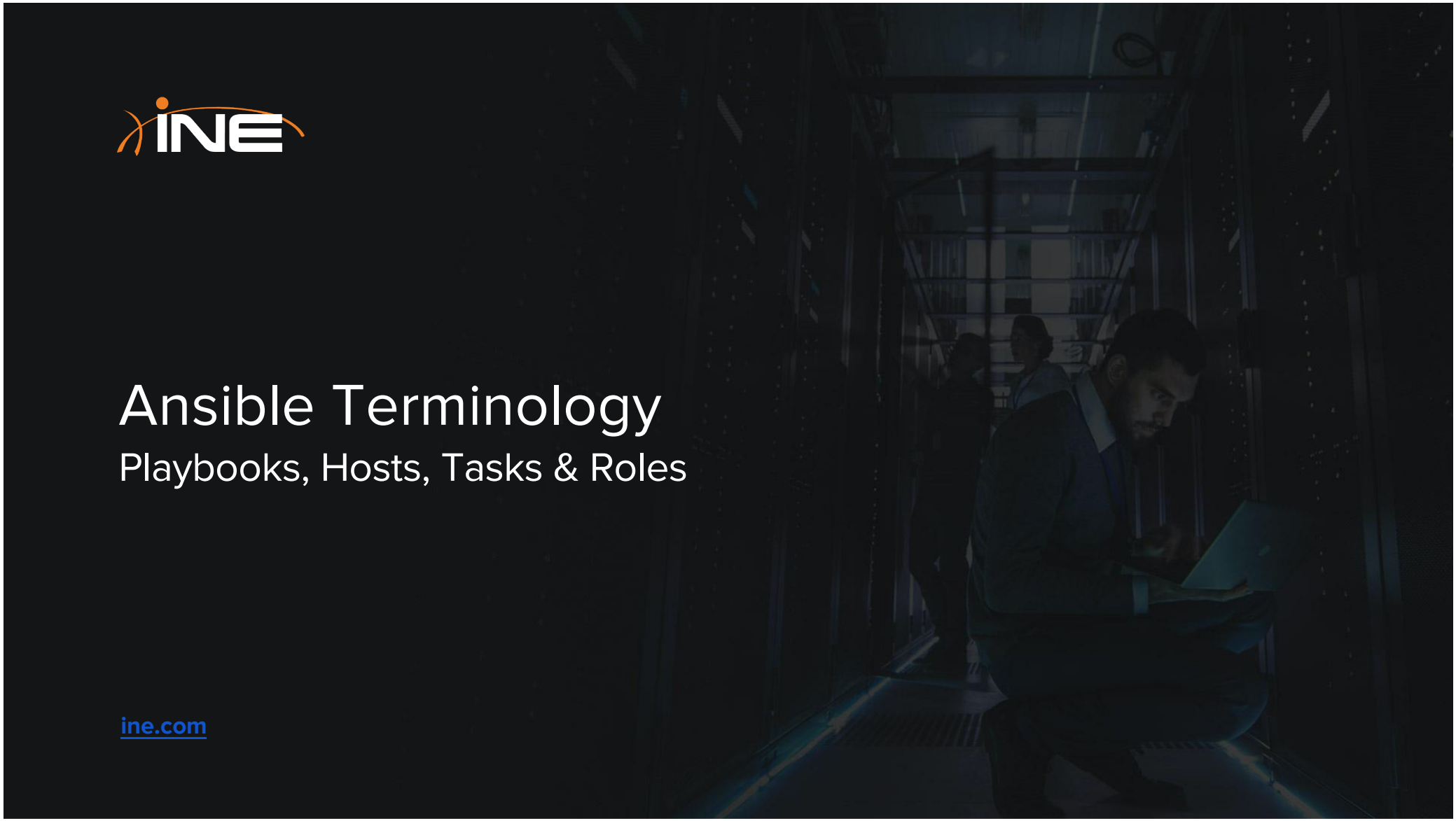
- Sometimes modules don’t yet exist for your particular need
  - For example, there is no module within the cisco.ios collection for obfuscating QoS commands.
  - In this case, there is usually a module created with the word, “command” in it that allows you to push whatever commands you want to the device.
    - ios_command module within cisco.ios
    - junos_command within juniper

Modules

- To view the modules that came with your ansible package:
  - ansible-doc -l ## “l” as in “Lion”
  - ansible-doc -l | grep ^ios

- So…how do I obtain/download my needed Ansible module?

Thanks for Watching!



Ansible Terminology

Playbooks, Hosts, Tasks & Roles

ine.com

Topic Overview

+ Playbooks
+ Hosts
+ Groups
+ Tasks
+ Roles

Playbooks

- Playbook : A collection of plays-and-tasks to be performed in a certain order
- Within a playbook, the minimum things that need to be defined are:
  - The hosts (or group) that will be targeted
  - The connection method
  - The name(s) of the collections and modules that will be used
  - A named, ordered listing of tasks (called “plays”)
- Playbooks are created using a syntax called YAML

Hosts & Groups

- Host is an individual machine (like a router or server) that can be targeted by Ansible
- If multiple hosts will be treated the same way then you can put then into named “groups” and reference that group in your Playbook
  - One can have groups of groups if desired
- Hosts are stored in a file called the “hosts” file
  - Typically located in the /etc/ansible/ directory

Tasks

- Task = single action (similar to a single IOS command) that results in “ok”, “change” or “failed”
- A task can be implemented as a one-time action using the Ansible CLI (called, “ad hoc”), or it can be part of a larger automated package within a Playbook.
- When included in a Playbook one-or-more tasks are grouped under a “play”

Task Operation

- By default, Ansible executes each task in order, one at a time, against all machines matched by the host pattern.
- Each task executes a module with specific arguments.
- When a task has executed on all target machines, Ansible moves on to the next task.
- If a task fails on a host, Ansible takes that host out of the rotation for the rest of the playbook.

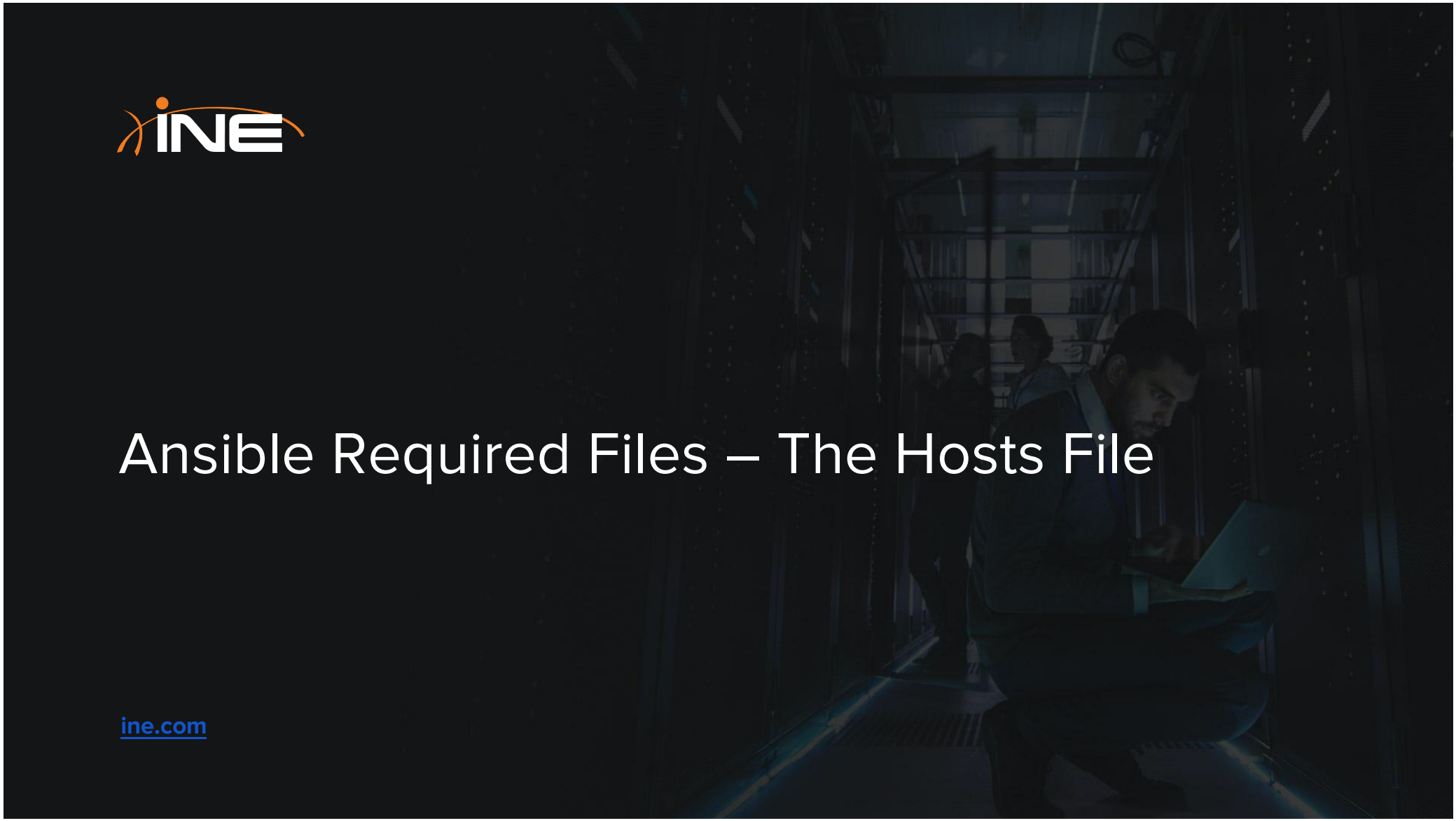
Roles

- "*A role enables the sharing and reuse of Ansible tasks. It contains Ansible playbook tasks, plus all the supporting files, variables, templates, and handlers needed to run the tasks. A role is a complete unit of automation that can be reused and shared.*" – Ansible documentation

Searching For Roles

- Roles are stored within the Ansible Galaxy (central repository)
- One can search for available roles using:
  - sudo ansible-galaxy search <keyword>
- One can view details of a role using:
  - sudo ansible-galaxy info <role-name>
- Installing a role:
  - sudo ansible-galaxy install <role>

Thanks for Watching!



Ansible Required Files – The Hosts File

ine.com

Topic Overview

+ Purpose Of Ansible Hosts File
+ Creating Your Own Hosts File

Hosts File: Purpose

- Required by Ansible in most situations
  - Hosts can be defined in Playbooks and Ad-Hoc commands, but this is not scalable
- Ansible assumes a default location of:
  - /etc/ansible/hosts
- Other locations for the “hosts” file can be referenced in the ansible.cfg file
- Can be created in .ini or .yaml format

Common Host Variables

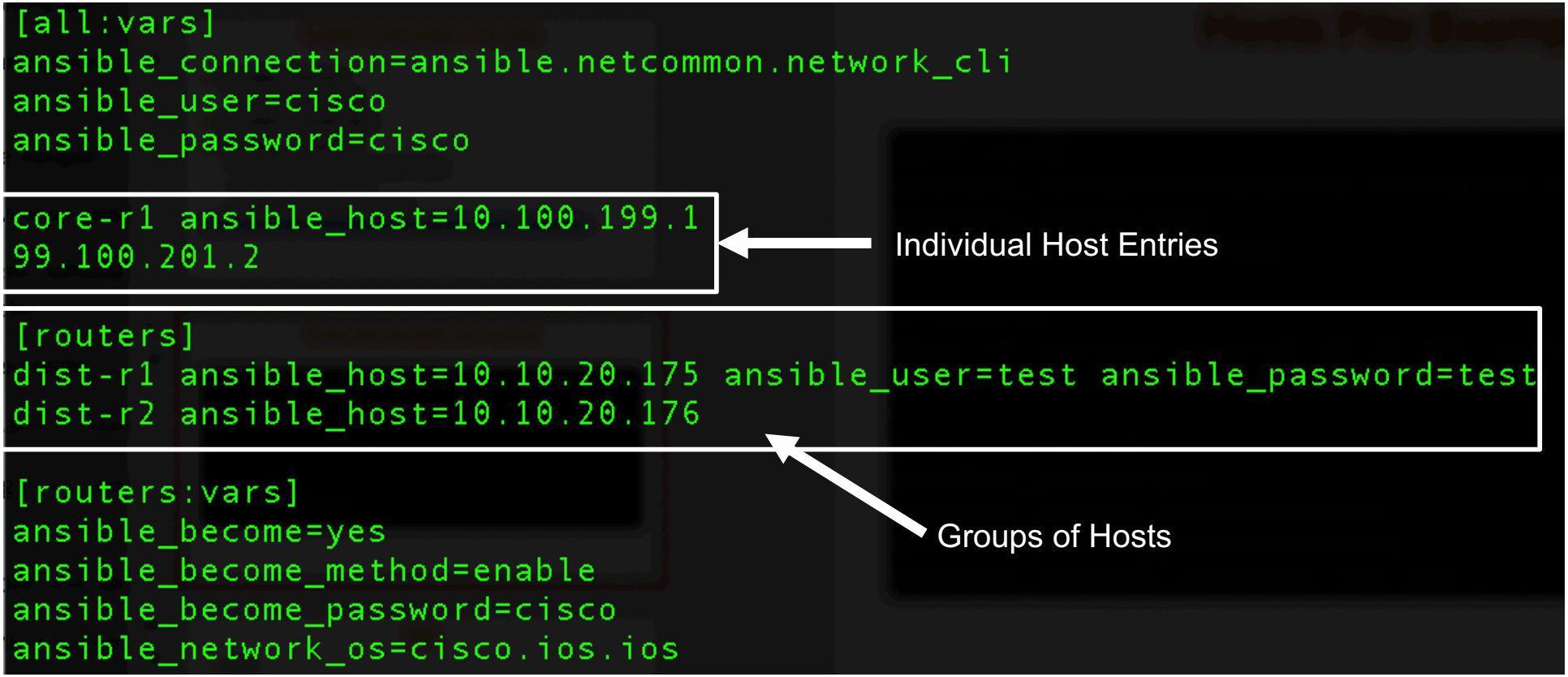
- Host variables instruct Ansible how to interact with hosts:
  - Connection type: SSH, HTTP, etc
  - SSH username
  - SSH password
  - Timeout values
  - MANY many other things
- Variables can be applied to:
  - Individual host entries
  - Groups
  - Groups of groups
  - All hosts and groups
- https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

Ansible Terminology & Concepts

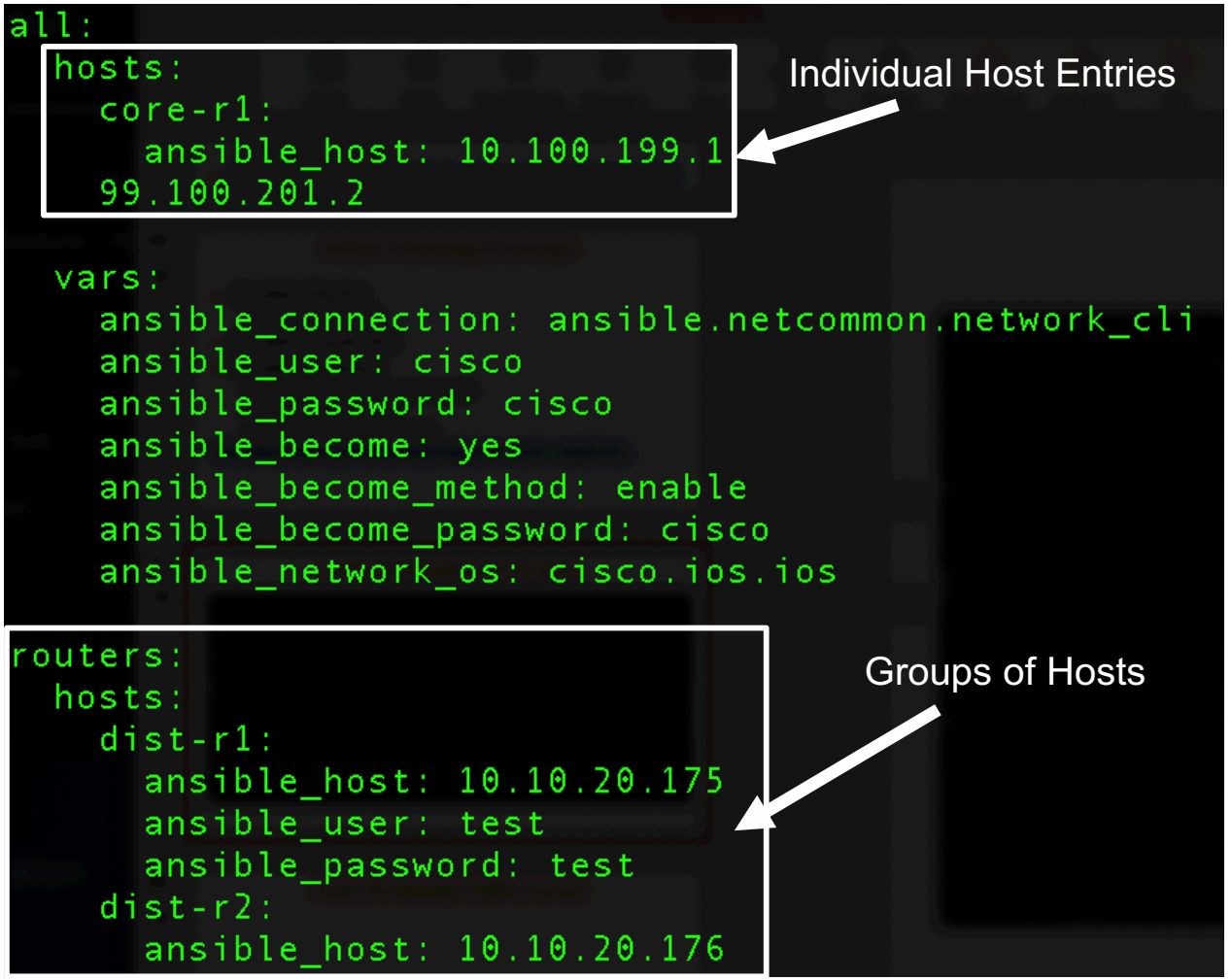
- Ansible Master
  - Aka “Control Node”
- Ansible Playbooks
- Ansible Modules
  - Make API calls to managed nodes
  - Apply configurations
  - Example: ios_config module

https://docs.ansible.com/ansible/latest/modules/ios_config_module.html

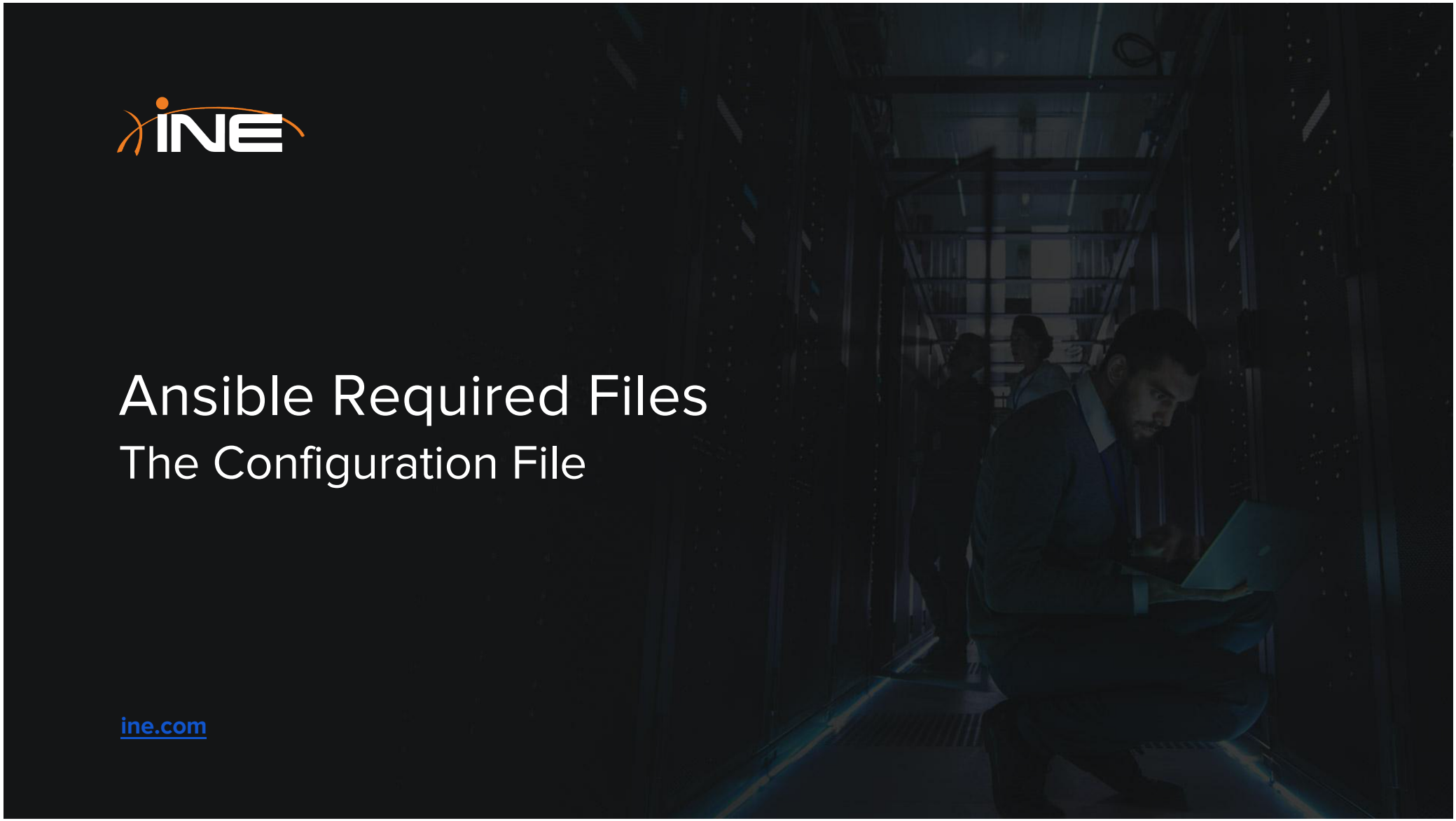
Hosts File Example (.ini Format)



Hosts File Example (YAML Format)



Thanks for Watching!



Ansible Required Files

The Configuration File

ine.com

Topic Overview

+ Purpose Of Ansible Configuration File
+ Commonly Made Edits

Ansible.cfg: Purpose

- Many default behaviors govern how Ansible works and interacts with hosts
- These defaults can be changed by editing the ansible.cfg file
- Default location: /etc/ansible
- Let's take a look at one!

Common Edits

- Remote_port = 22
  - Modify if hosts are listening on non-standard SSH port
- Gathering = explicit
  - Many playbooks don’t require gathering of facts
- Host_key_checking = False
  - Disables SSH host key checking

Common Edits

- Edit [privilege_escalation] variables if all hosts use same method:
  - Become = True
  - Become_method=enable
  - Become_user=<enable password>
- Host_key_auto_add=true
  - Automatically adds new host keys without pausing and prompting you

Thanks for Watching!

An Overview Of YAML

ine.com

Topic Overview

+ Introduction To YAML
+ Basic Syntax Rules
+ YAML Lint
+ Long Strings
+ Mappings-Dictionaries
+ Lists
+ Indentation
+ YAML Exercise

YAML

- YAML = “YAML Ain’t Markup Language”
- Data serialization language
- Designed to be “human friendly”
- Used heavily in Ansible for:
  - Creation of Playbooks
  - Creation of Hosts files
  - Roles
  - Etc
- YAML Documentation:
  - https://yaml.org/spec/1.2/spec.html

Basic Syntax Rules

- From the YAML documentation:
  - Indentation may be used for structure
  - Colons separate key: value pairs
  - Dashes (i.e. hyphens) are used to create “bullet” lists.
- Examples:

```
american:
  - Boston Red Sox
  - Detroit Tigers
  - New York Yankees
national:
  - New York Mets
  - Chicago Cubs
  - Atlanta Braves
```

```
routers:
  hosts:
    dist-r1:
      ansible_host: 10.10.20.175
      ansible_user: test
      ansible_password: test
    dist-r2:
      ansible_host: 10.10.20.176
```

YAML Lint

- http://www.yamllint.com/
- Website that helps validate documents for correct YAML syntax
  - Did you use correct indentation?
  - Did you forget to add hyphens to your list?
  - Did you forget a colon anywhere?
- Does NOT validate against application rules

Long Strings

- Long strings of characters can be written on separate lines in YAML
- Example:

```
"This is my long string of text that just keeps going and going and going!"
```

is exactly the same as…

```
"This is my long
 string of text
 that just keeps
 going and going
 and going!"
```

Mappings/Dictionaries

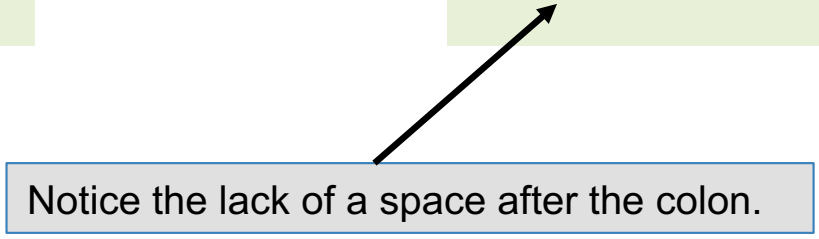
- YAML mappings are used to add definitions to objects
- Mappings are used to create dictionaries
- Mappings are indicated by appending a colon-and-space after an object:

Correct

```
Bob: Payroll
Larry: Engineering
Sue: Executives
```

Incorrect

```
Bob: Payroll
Larry: Engineering
Sue:Executives
```



YAML Lists

- Lists are dictionary items (i.e. mappings) that may contain two or more definitions
- List items begin with a hyphen-and-a-space
- If each list item is a unique string, must be preceded by a hyphen-space
- If each list item is a mapping/sub-list a single hyphen-space can be used to delineate them.

YAML List Examples

- A dictionary/list named “My_Friends” with each list item being a string:

Correct

```
My_Friends:
      - Bob
      - Larry
      - Sue
```

Incorrect

```
My_Friends:
      - Bob
        Larry
      - Sue
```

This will generate an error.

Incorrect

```
My_Friends:
      - Bob
        Larry
      - Sue
```

…will be interpreted as…

```
My_friends:
  - "Bob Larry"
  - Sue
```

YAML List Examples

- A dictionary/list named “Departments” with each list item being a sub-list of additional dictionaries:

Correct

```
Departments:
    - Payroll:
      - Dept_number: 100
      - Building: 2
    - Engineering:
      - Dept_number: 84
      - Building: 1
```

Notice the levels of indentation!

Also Correct

```
Departments:
    -
      Payroll:
      -
        Dept_number: 100
        Building: 2
      Engineering:
      -
        Dept_number: 84
        Building: 1
```

Indentation

- Indentation in YAML is used to signify ownership or parent-child relationships
- Do NOT use the tab key…use the spacebar to add indentation
- Indentation can be of any amount as long as it is consistent

Correct

```
- Bob:
    dept: engineering
- Larry:
    dept: payroll
    personality: hateful
```

Incorrect

```
- Bob:
    dept: engineering
- Larry:
    dept: payroll
      personality: hateful
```

"personality" is NOT a child of "dept". It is at the same level and should be indented at the same level.

YAML Exercise

- Use the YAML lint website to create/validate a YAML document that:
  1. Creates a dictionary/mapping called, “Routing_Protocols”
  2. ”Routing_Protocols” is mapped to a list of items (actual Routing Protocol names) each of which is another dictionary:
     - BGP
     - OSPF
     - EIGRP
  3. For each routing protocol (BGP, OSPF, EIGRP) add the following two list items:
     1. Process_ID = integer (pick whatever integer you wish)
     2. Process_name = string (pick whatever string you wish)

YAML Exercise Example

- The following is NOT in correct YAML syntax...make it so!
  I. Routing_Protocols
     A. BGP
        i. Process_Id = 100
        ii. Process_name = INE
     B. OSPF
        i. Process_Id = 45
        ii. Process_name = Cisco
     C. EIGRP
        i. Process_Id = 4
        ii. Process_name = Ansible

YAML Exercise Solution

YAML Lint

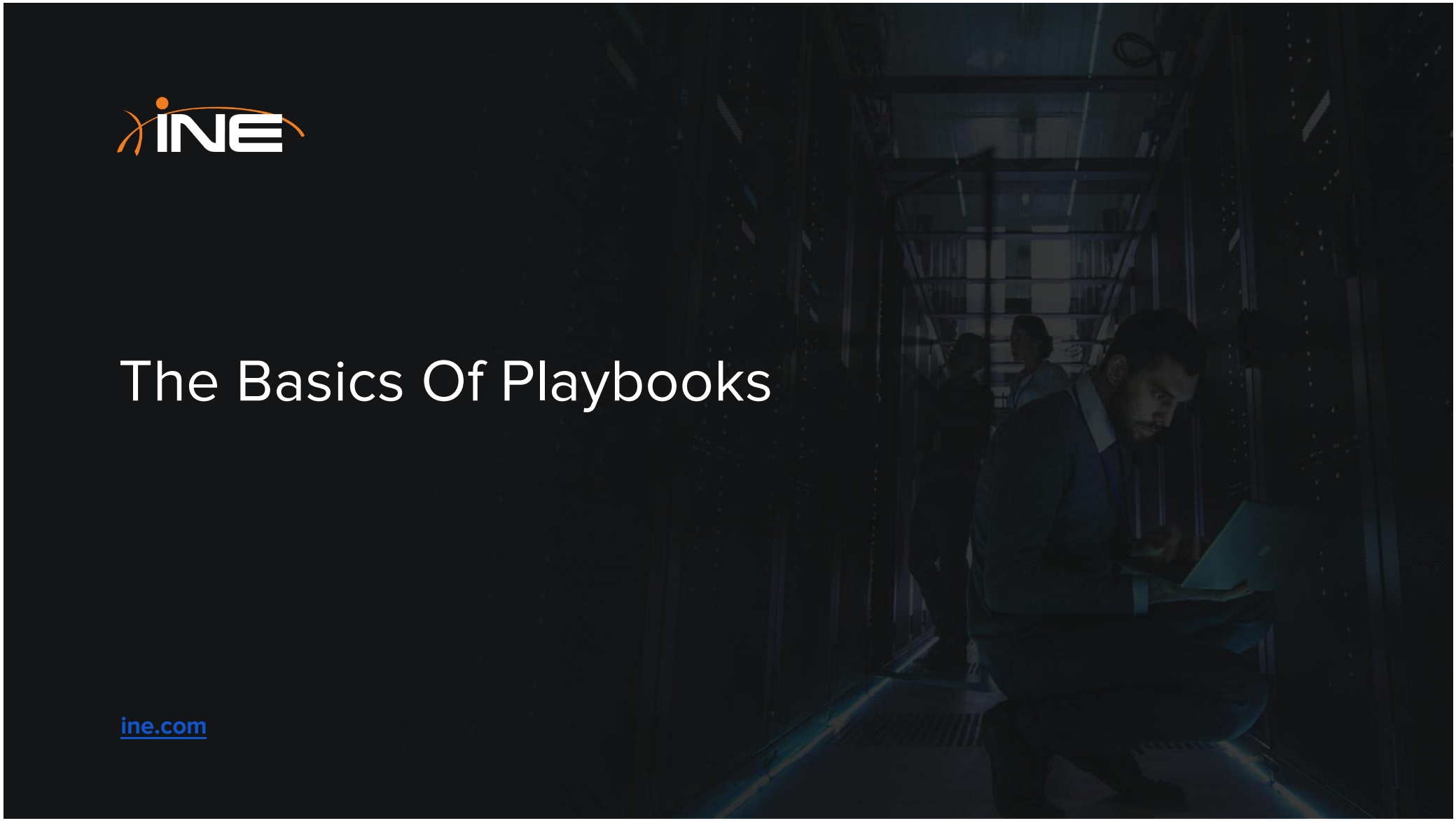
Paste in your YAML and click "Go" - we'll tell you if it's valid or not, and give you a nice clean UTF-8 version of it. Optimized for Ruby.

```
---
Routing_Protocols:
  -
    BGP:
      -
        Process_Id: 100
        Process_name: INE
    EIGRP:
      -
        Process_Id: 4
        Process_name: Ansible
    OSPF:
      -
        Process_Id: 45
        Process_name: Cisco
```

Go

Valid YAML!

Thanks for Watching!



The Basics Of Playbooks

ine.com

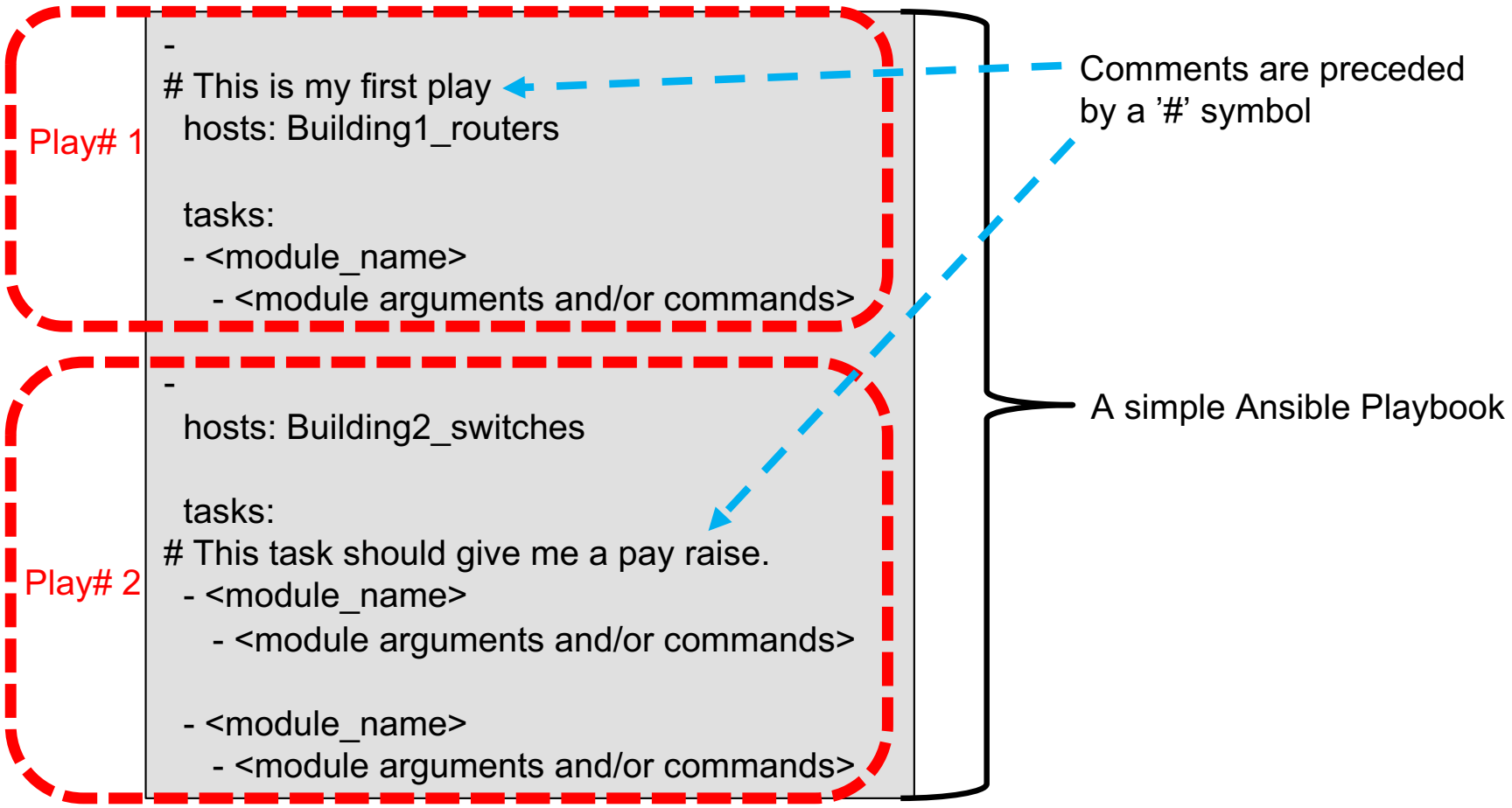
Topic Overview

+ Playbook Overview
+ Playbook Prerequisites
+ Playbook Design

Playbook Overview

- Documents created in YAML (.yaml or .yml)
- Allow Ansible to perform multiple tasks against multiple hosts
- A playbook runs one-or-more “plays” in an ordered list against a targeted group of hosts
  - A play = collection of one-or-more tasks
- At minimum a Play must specify:
  - The managed nodes to target, using a pattern
  - At least one task to execute
- Descriptive comments can be added anywhere in a playbook by preceding them with a hash/octothorp symbol.

Playbook Overview



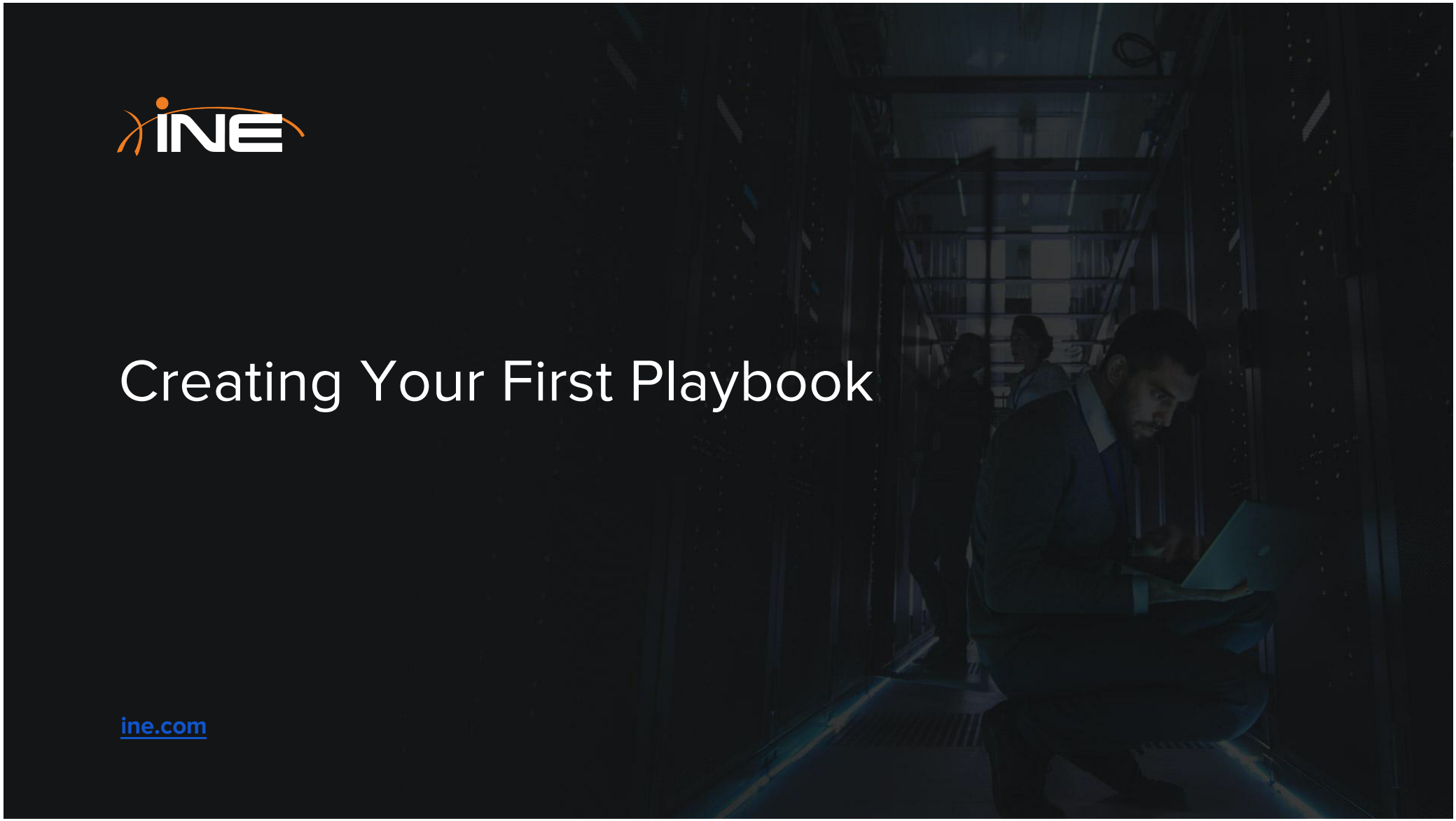
Playbook Prerequisites

- Before creating a Playbook a few things should be considered:
  - Do you have an Ansible “hosts” file that provides necessary reachability parameters?
  - Have you downloaded the Ansible collection containing the modules your Playbook will be referencing?
  - Have you confirmed SSH reachability to your hosts?
  - Will you need Ansible to “Gather_Facts”?

Playbook Design

- Identify your objectives for each task within a play.
  - Is your objective dependent upon successfully completing a prior objective?
  - Will completing your objective require that Ansible answer any automated prompts from your targeted host(s)?
- Can your objective be met by utilizing an Ansible module?
- When running your Playbook for the first time, run against a single host.

Thanks for Watching!



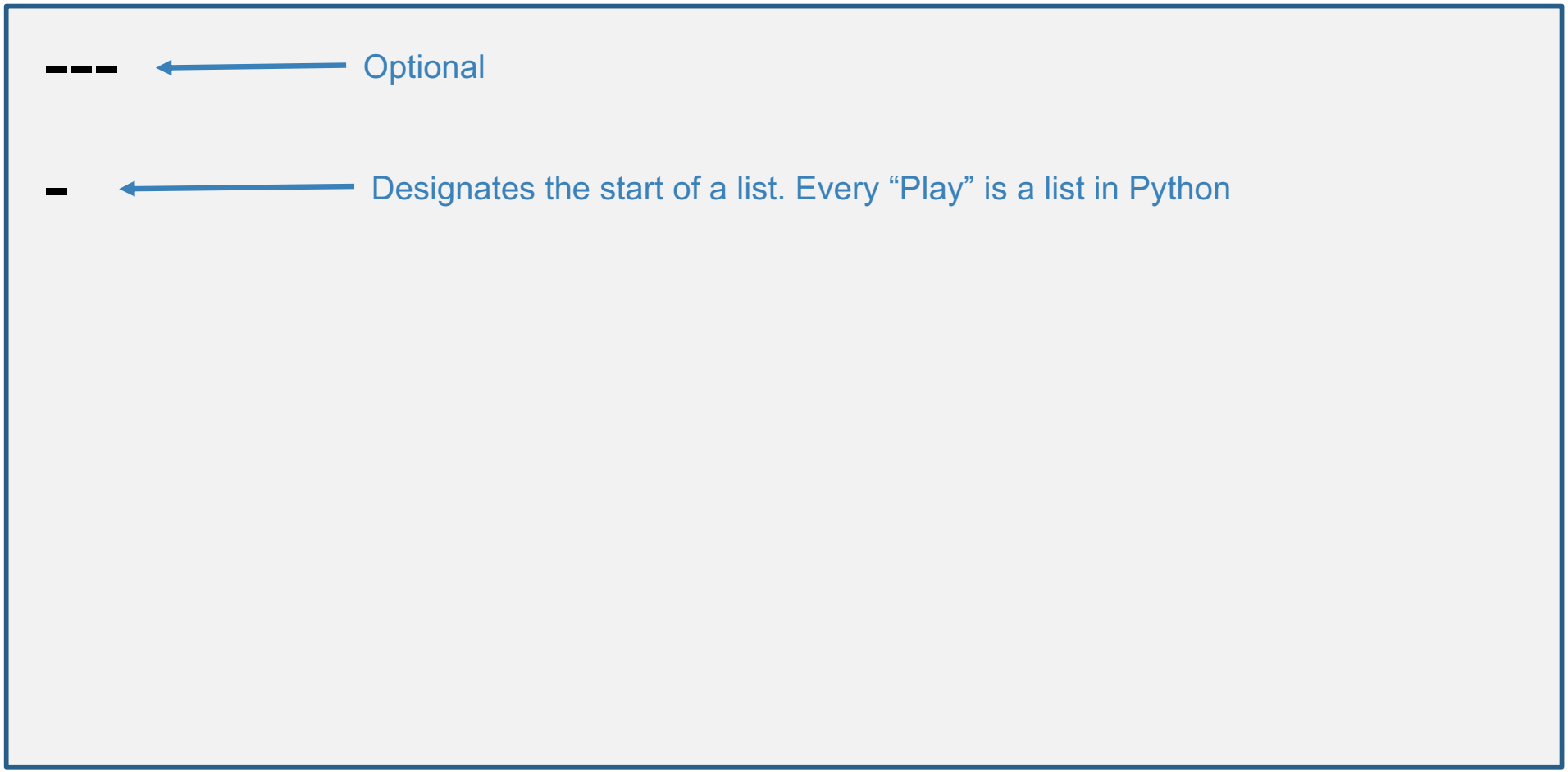
Creating Your First Playbook

ine.com

Topic Overview

+ Essential Playbook Components
+ Using a “Command” module
+ Interpreting Ansible Module Documentation
+ Utilizing The “Debug” Module

Essential Playbook Components



Essential Playbook Components

```
---

-
    name: Description of my first play…
    hosts: Building_1_routers
    gather_facts: no
```

The first list-item may begin at the same level as the hyphen, or one line below it…but it MUST be indented from the hyphen-space.

Required

Optional

Essential Playbook Components

```
---

-
    name: Description of my first play…
    hosts: Building_1_routers
    gather_facts: no

    <A line-break is not required here…but makes the playbook look nicer.>

    tasks:
    -
```

Required. Each play must have at least one task.
A task begins a new sub-list composed of an Ansible module and one-or-more arguments.

The level of indentation for this hyphen-space doesn't matter as long as it is at the same indentation-level (or greater) than the "t" in "tasks:".

Essential Playbook Components

```
---

- name: Description of my first play…

  hosts: Building_1_routers
  gather_facts: no

  tasks:

  -
    name: This is my first task and accomplishes …
    <module_name>:
      <arguments>
```

Optional

Required

(Required) The level of indentation for the first argument/command of the module must be greater than the name of the module itself on the line above.

Using “Command” Modules

- Most vendor collections for Ansible contain “command” modules.
- Useful for sending one-or-more CLI commands to your devices over an SSH connection.
- Let’s look at a Playbook using one of these modules…

Registering Return Values

- By default, Ansible will not display anything the host sends back to you over the SSH connection unless it results in a task error.
- One can “register” the return value or values of a task as a new (named) variable

```
- name: This is my task
  module:
      - show this
      - show that
  register: my_output
```

This instructs Python to create a new variable called, “my_output” and place within that variable all output (i.e. “return values”) that resulted from this task.

Using The Debug Module

- The “debug” module is included with all Ansible distributions
- It is used to print text to your screen
- This module is run as its own, separate task.

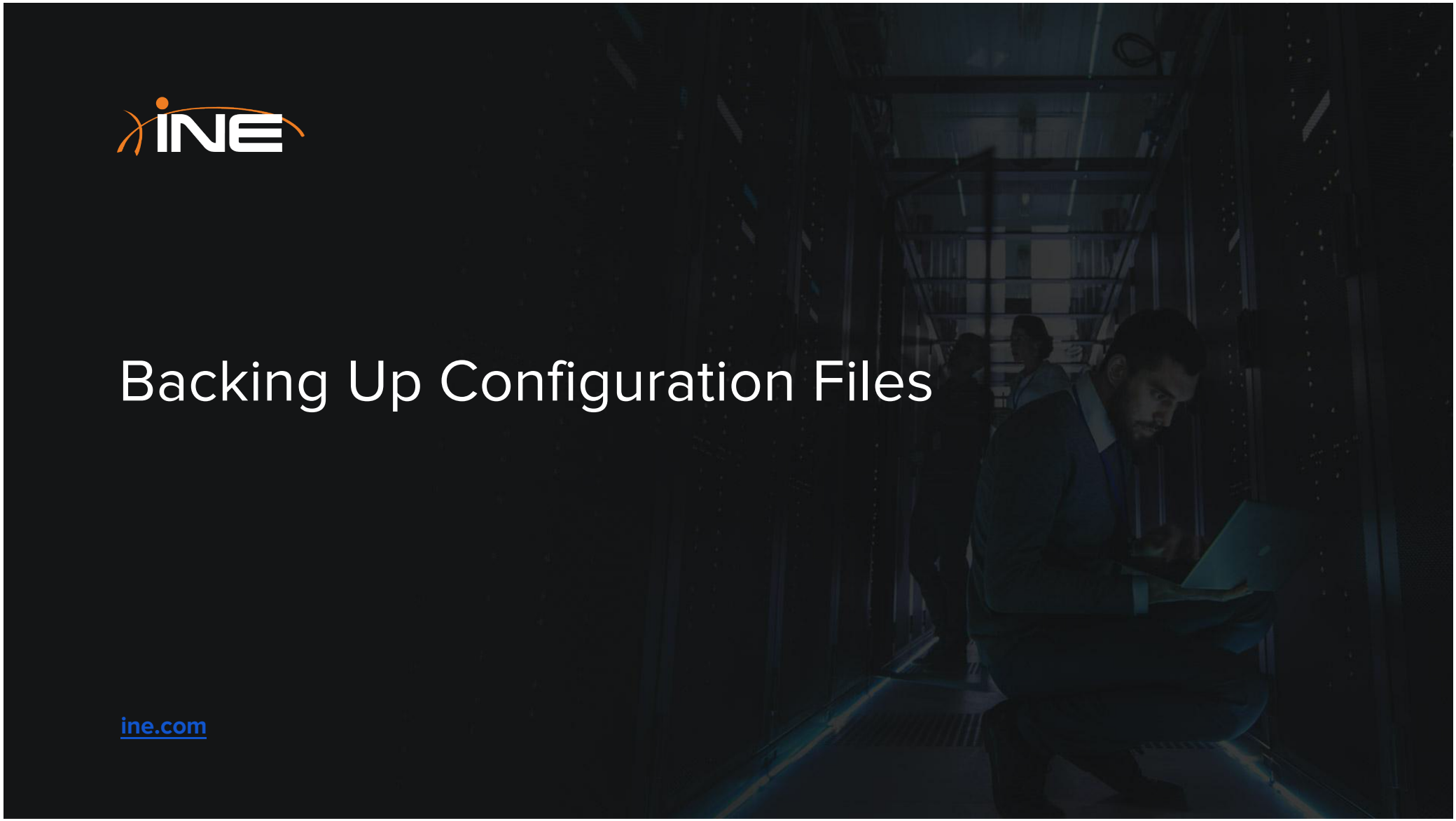
```
- name: This is my task
  module:
      - show this
      - show that
  register: my_output

- debug: var=my_output.stdout_lines
```

Responding To Prompts

- Let’s see what the IOS_Command module documentation has to say about responding to prompts…

Thanks for Watching!



Backing Up Configuration Files

ine.com

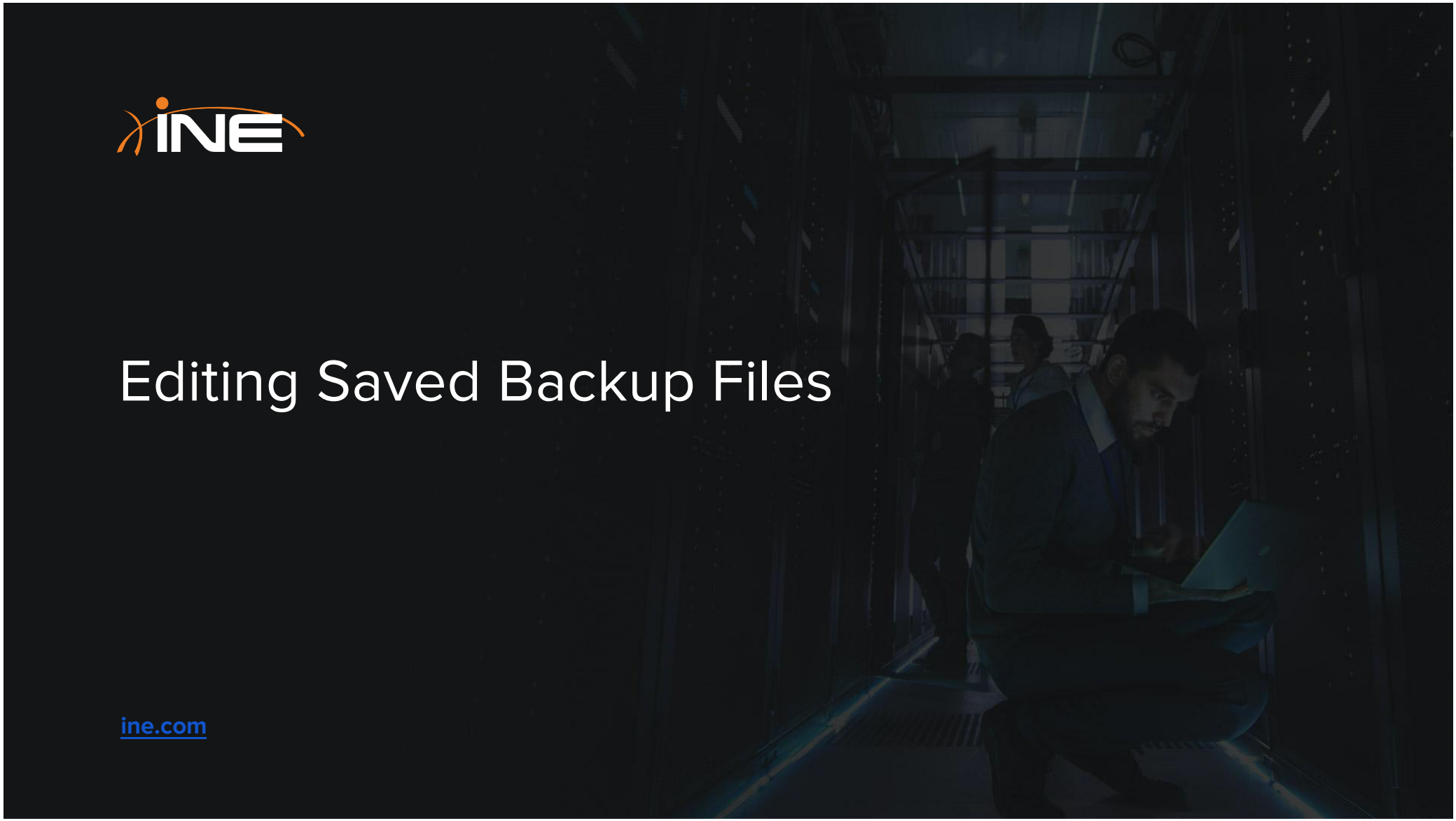
Topic Overview

+ Backup Up Configuration Files

Backup Of Configs

- A common use of Ansible is to store copies of “golden configs”.
- No need to have external FTP, TFTP, SCP, etc server for this purpose.
- More-than-one module could accomplish this objective, but we’ll be using the cisco.ios.ios_config module
- Various caveats exist

Thanks for Watching!



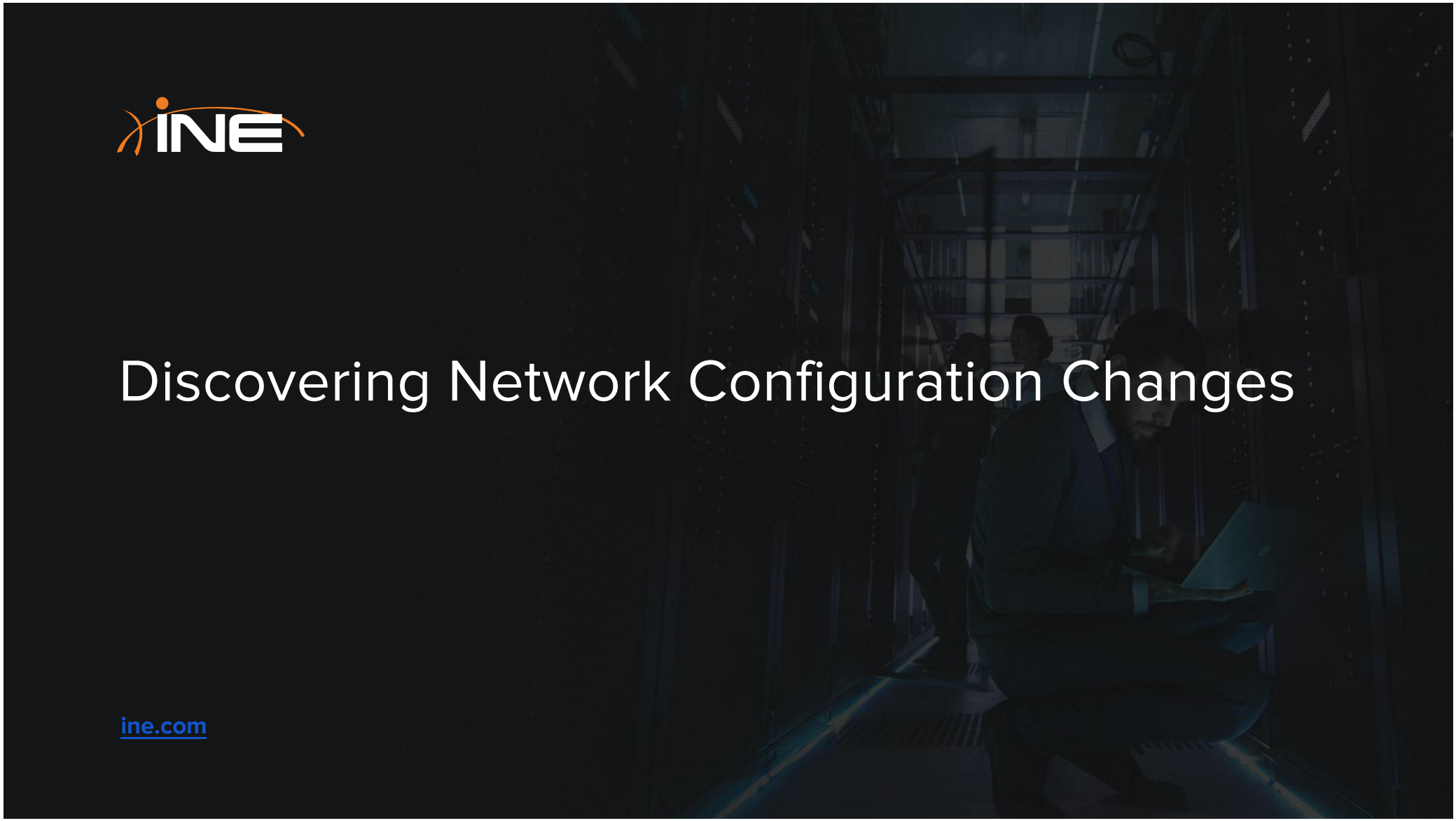
Editing Saved Backup Files

ine.com

Topic Overview

+ Removing Lines From Saved, Backup Up Configuration Files

Thanks for Watching!



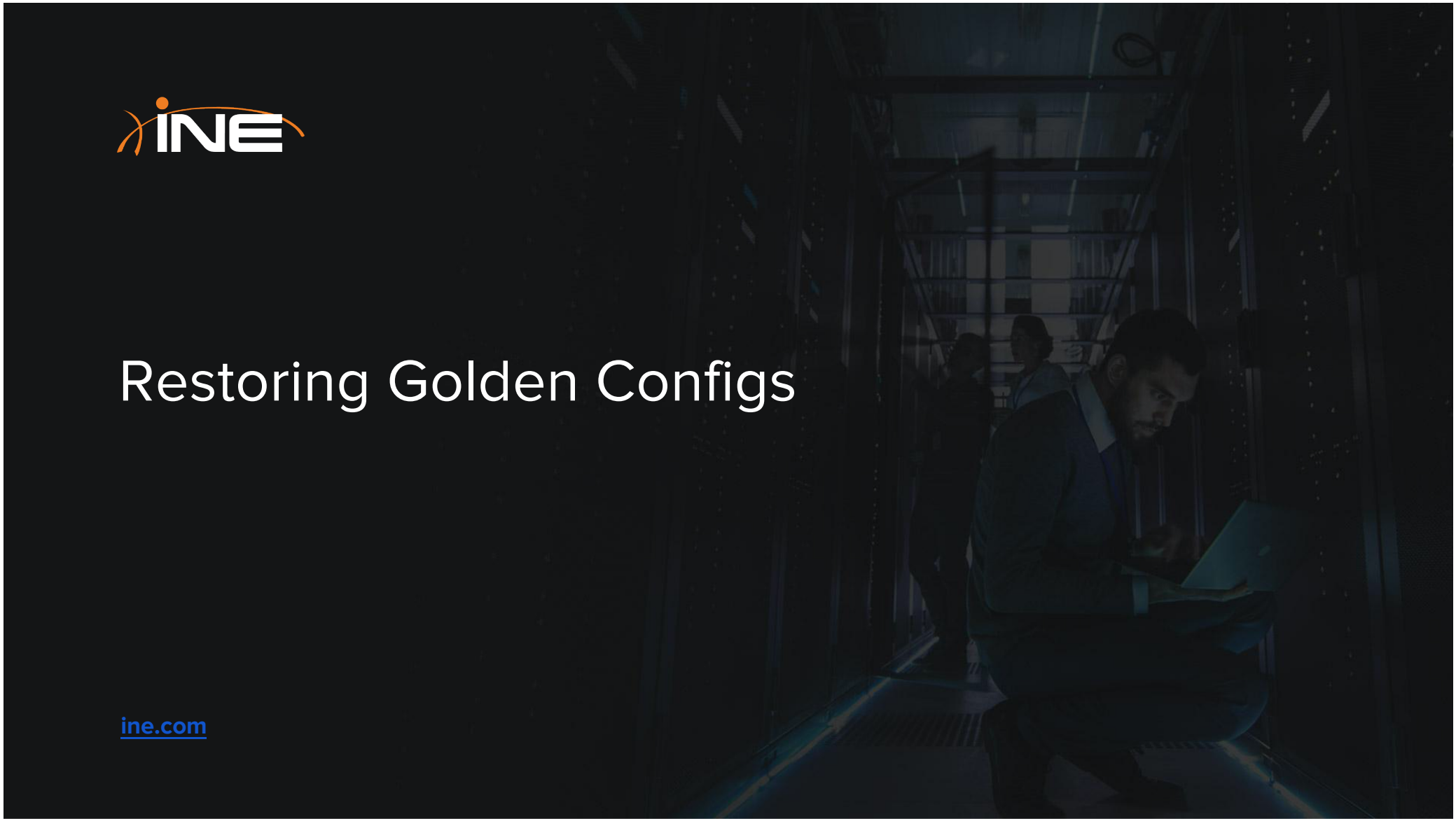
Discovering Network Configuration Changes

ine.com

Topic Overview

+ Using Ansible To Scan For Network Nodes With Changed Configurations

Thanks for Watching!



Restoring Golden Configs

ine.com

Topic Overview

+ Restoring Golden Configs On Network Devices

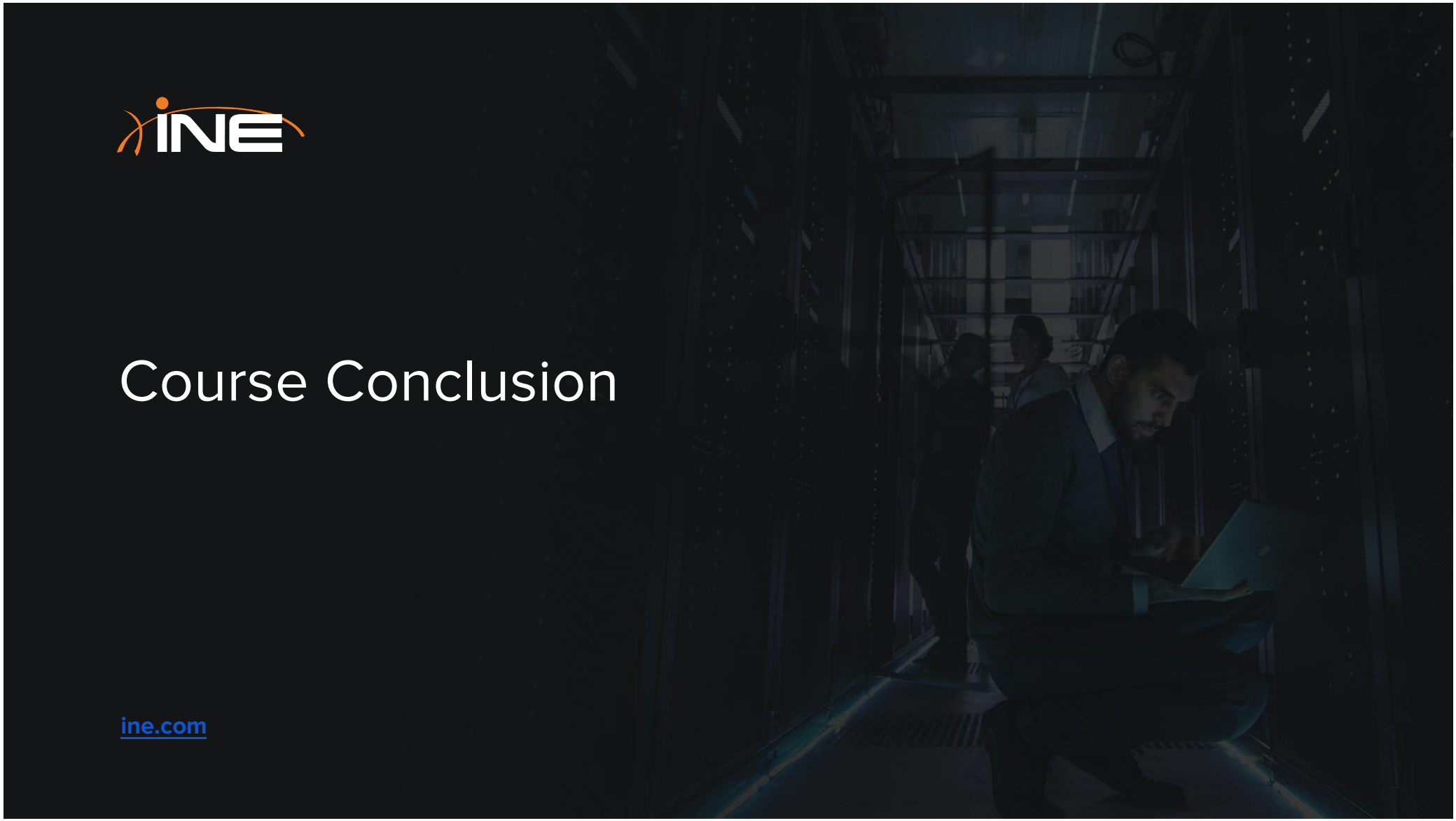
Restoring Configs

- After discovery of an unauthorized configuration change on a device(s) you may wish to restore that device back to its authorized ”Golden Config”.
- Methods available to accomplish this:
  - Use an Ansible module to push individual commands one-by-one
  - Use an Ansible module to rewrite entire configuration sections (i.e. “ios_ospfv2” module)
  - Use a combination of modules to replace entire (i.e unauthorized) configuration

Method Caveats

- The following demonstrations accomplishes our objective but with these caveats:
  - This method is best used when replacing entire configurations on one or two devices…not dozens or hundreds
  - The Ansible “Command” built-in module seems to ignore the Ansible Hosts file and instead refers to the local system’s “Hosts” file
  - The Ansible “Command” module logs into the remote system (i.e. router or switch) utilizing the local username of the Ansible Control Node
- Remember, Ansible utilizes SSH. If the unauthorized configuration change(s) disabled SSH, Ansible CANNOT be used to restore the remote configuration file.

Thanks for Watching!



Course Conclusion

ine.com

Thanks for Watching!