INTRODUCTION TO ANSIBLE FOR NEWBIES

Christoph Stoettner <stoeps@vegardit.com>

WHO AM I?

- Christoph Stoettner
- Senior Consultant @Vegard IT
- Focusing on HCL Connections deployments and migrations
- Ansible since 2017 — Social Connections 12 [1]

Example code on github.com/stoeps13/ansible-examples

1. share.stoeps.de/2017-10-16-ansible4connections.pdf

HANDCRAFTED SERVERS

- Hard to maintain
- Setups are not reproducible
- Complicated vendor documentation
- Inhouse documentation outdated

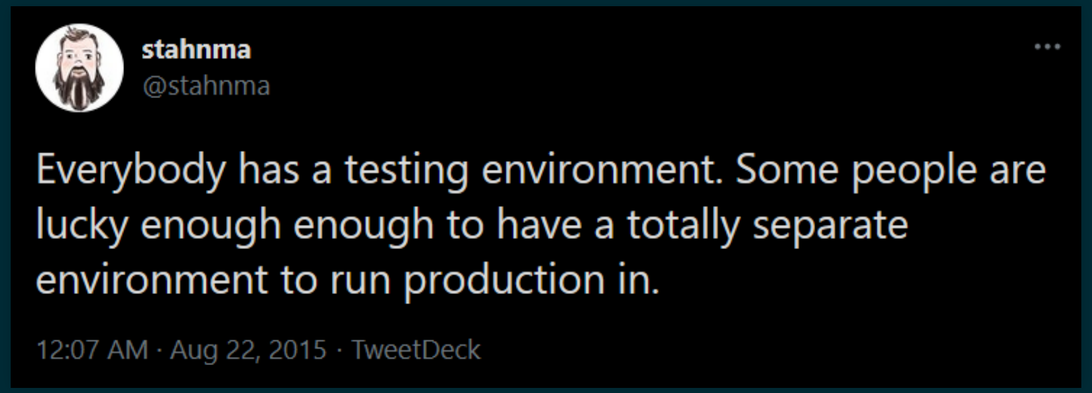
IMMUTABLE VERSUS MUTABLE SERVER

- Mutable infrastructure just gets updates
  - Software 6.0 → + Ifix 1 + Ifix 2 + Ifix 3
  - In production 6.0 → ifix 3
  - Result will be different
- Immutable creates a new environment with 6.0.x
  - Migrates data after testing

SNOWFLAKE SERVERS

- Special tweaks or versions needed for proper function
- Exception of your standards
- Difficult to reproduce
- Fragile if they need a change

TEST ENVIRONMENTS



twitter.com/stahnma/status/634849376343429120

WHY ARE DEDICATED TESTENVIRONMENTS IMPORTANT?

- Reliable testing can give you confidence during live migration
- Applying Fix 3 over Fix 2 over Fix 1 often different from Fix3 over Fix1
- Use the same scripts to build development, test or production systems
- Handcrafted is always different from production

ADVANTAGES

- Developer
  - Build a development environment which is compareable to production
- Adminstrator
  - Build a test environment to go through a migration

BE AS PRECISE AS POSSIBLE

- Avoid different hostnames
  - Production: example.com
  - Test: test.example.com
- Better:
  - example.com
  - example-test.com

HOW CAN WE SOLVE THIS?

- Deployment **and** Application development should follow a fully automated approach
- Avoid Snowflakes
- Easier to have a full clone of production as test environment
- Reducing production bugs caused by configuration differences

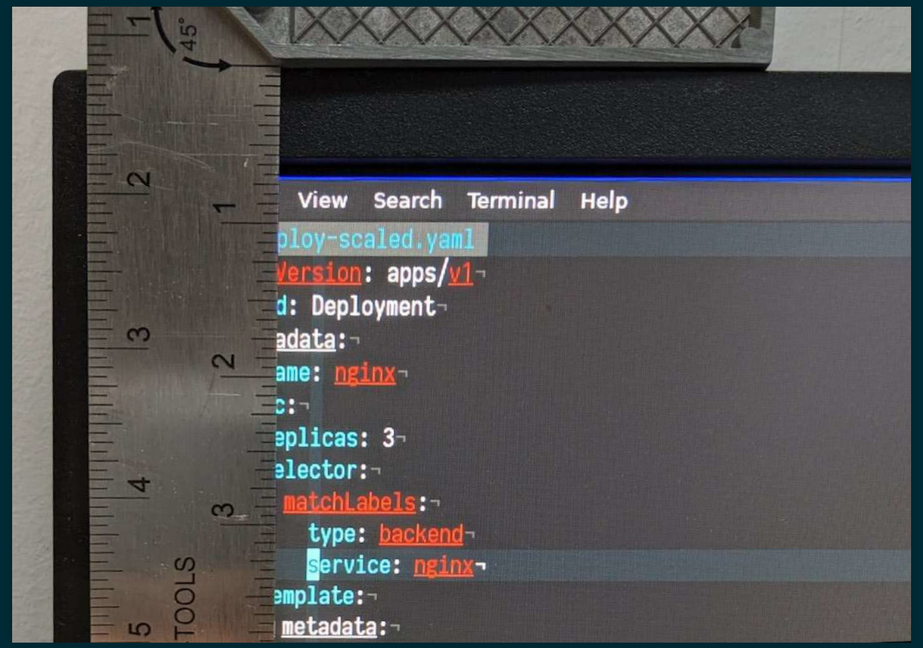
AUTOMATE DEPLOYMENTS AND CONFIGURATION CHANGES

- Large ecosystem of tools to do automatic deployments
  - Wikipedia OSS Configuration Management
- Puppet puppet.com
- Chef www.chef.io
- Saltstack saltstack.com
- Ansible ansible.com

ANSIBLE

- Written in Python
- Encryption and Security built in
- Easy to read (Everything is YAML)
- Easy to use (Extensible via modules)
  - Uses SSH

YAML Tool Kit



ANSIBLE HISTORY

- Created by AnsibleWorks Inc, acquired by Red Hat in 2015
- Initial release: 20. February 2012
- Stable release: 2.10.6
- 3.0.0 announced for the 16th of February (two days ago)

A VERY IMPORTANT TERM: IDEMPOTENCY

Mathematics

denoting an element of a set which is unchanged in value when multiplied or otherwise operated on by itself

IDEMPOTENCY — EXAMPLE

- Add entry to hosts
  - Don’t add when present
  - Change if different
- Restart services only when changes were made

Not idempotent

```
echo "192.168.1.1 cnx-websphere.example.com" >> /etc/hosts
```

Idempotent

```
grep -qxF '192.168.1.1 cnx-websphere.example.com' /etc/hosts || \
echo "192.168.1.1 cnx-websphere.example.com" >> /etc/hosts
```

WHAT IS ANSIBLE?

- Helps automating tasks during installation and migration
- Secure (SSH)
- Open (tons of free playbooks)
- Well documented

WHAT IS ANSIBLE NOT?

- A GUI Tool (Get used to console!)[1]
- A one click installer

1. Ansible Tower and AWX are browser tools

ANSIBLE INSTALLATION

- `pip install ansible` on the machine you want to run it
- Newer version than distribution package
- Needs internet connection
- targets need at least `ssh` and `python` installed

WINDOWS AND ANSIBLE

- Ansible "server" needs Linux (but works with WSL)
- Windows support through
  - Windows Remote Shell (WinRM)
  - SSH

INVENTORY INI OR YAML FORMAT

```ini
[leafs]
leaf01.example.com
leaf02.example.com

[spines]
spine01.example.com
spine02.example.com

[network:children]
leafs
spines
```

```yaml
---
leafs:
  hosts:
    leaf01.example.com:
    leaf02.example.com:

spines:
  hosts:
    spine01.example.com:
    spine02.example.com:

network:
  children:
    leafs:
    spines:
```

VARIABLES IN INVENTORIES

```
[leafs]
leaf01.example.com
leaf02.example.com

[leafs:vars]
username=abc
```

```
---
leafs:
  hosts:
    leaf01:
    leaf02:
  vars:
    username: abc
```

VARIABLES

- Lots of places to define
- Presedence mportant for large environments

no hyphens in variable names!

Allowed variable

```
ldap_user: abc
```

Not allowed variable

```
ldap-user: abc
```

1. command line values (for example, `-u my_user`, these are not variables)
2. role defaults (defined in role/defaults/main.yml) [1]
3. inventory file or script group vars [2]
4. inventory group_vars/all [3]
5. playbook group_vars/all [3]
6. inventory group_vars/* [3]
7. playbook group_vars/* [3]
8. inventory file or script host vars [2]
9. inventory host_vars/* [3]
10. playbook host_vars/* [3]
11. host facts / cached set_facts [4]
12. play vars
13. play vars_prompt
14. play vars_files
15. role vars (defined in role/vars/main.yml)
16. block vars (only for tasks in block)
17. task vars (only for the task)
18. include_vars
19. set_facts / registered vars
20. role (and include_role) params
21. include params
22. extra vars (for example, `-e "user=my_user"`)(always win precedence)

docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html

PLAYBOOK

- Run commands (so called tasks) on your inventory servers
- Select servers or server groups
- Roles
- Tasks
- Handlers

TASKS

- Lots of modules built-in
  - Package install
  - Copy and Edit files
  - Create files and folders (directly and with templates)
  - Manage services
  - Command
  - Shell
- Sudo aware
  - Become: true
  - Become_user: xyz

TASKS FOR DIFFERENT OS

```
...
tasks:
  - name: Install mkpasswd
    package:  1
      name: whois
      state: present
    when: ansible_os_family == "Redhat"     2

  - name: Install mkpasswd
    package:  3
      name: expect
      state: present
    when: ansible_distribution == "Ubuntu"  4
```

1. or use `yum`
2. valid terms are `Redhat` | `Darwin` | `Debian` | `Windows`
3. or use `apt`
4. check OS family (Debian) or distribution

EXAMPLE (BUILD AN ANSIBLE ROLE)

- Most products of IBM or HCL need disabled SELinux during installation
- So let's disable SELinux on a host
- Additional steps will be
  - Configure `limits.conf`
  - Reboot after changes
  - Create a user
  - Install packages with `yum`
- All example files can be found at github.com/stoeps13/ansible-examples
  - Branches named for the steps

DISABLE SELINUX (INVENTORY)

inventory

```
[websphere_servers]
cnx-was.stoeps.internal (1)
```

(1) if hostname is resolvable that is enough

Sometimes you need to add IP or SSH Port! For example

```
[websphere_servers]
cnx-was.stoeps.internal ansible_host=10.0.11.101 ansible_port=2222
```

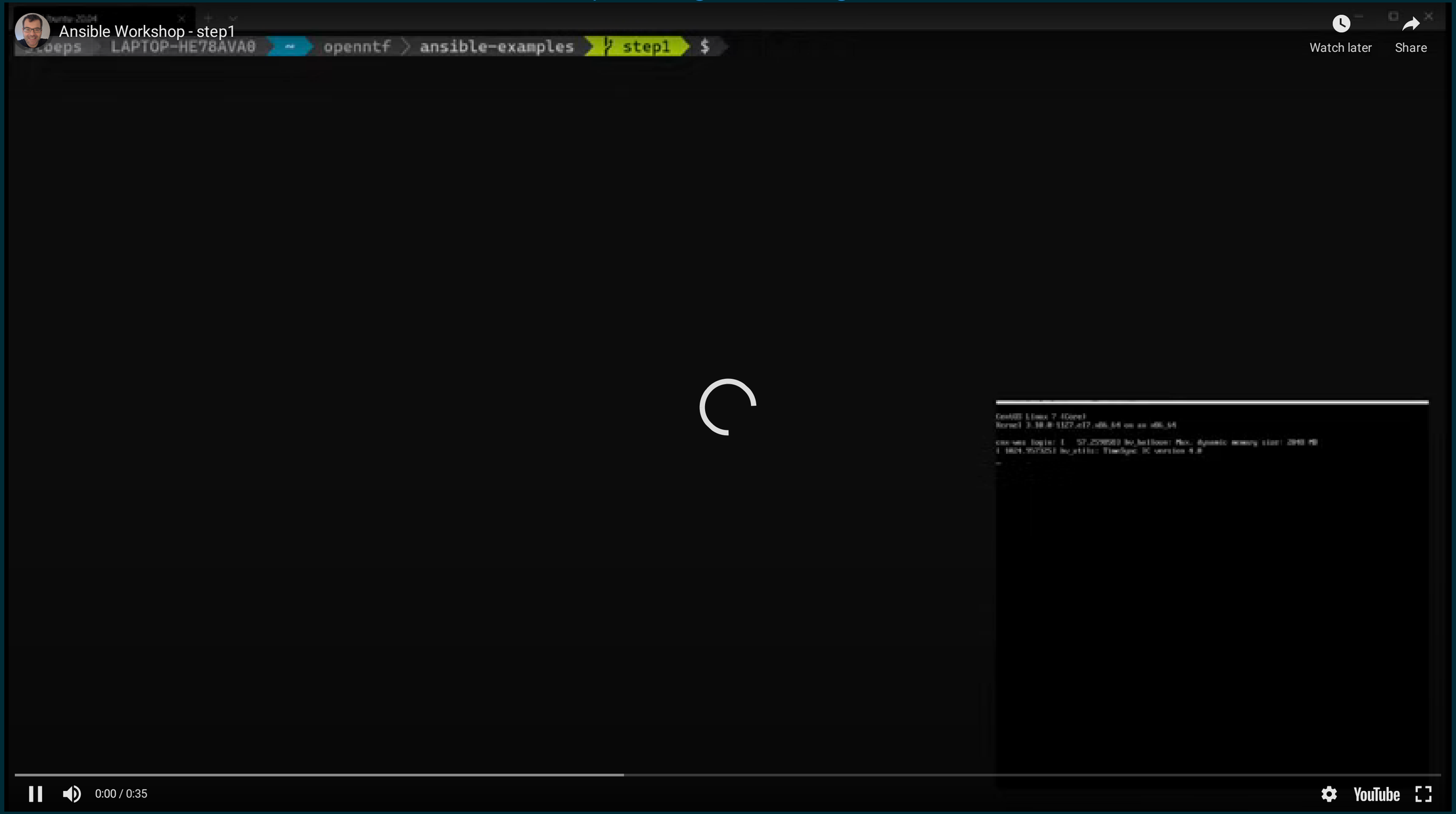
SET SELINUX TO `permissive`

playbook.yml

```yaml
---
- hosts: websphere_servers   1
  become: yes                2
  become_user: root          3
  tasks:                     4
    - name: ensure selinux is set to permissive
      selinux:               5
        policy: targeted
        state: permissive    6
```

1. Run this tasks on this server group
2. Use `sudo` to execute command
3. `sudo` to user `root`
4. tasks (one or multiple tasks)
5. use module `selinux`
6. `policy` and `state` are arguments / parameters for module `selinux`

youtu.be/g8OvWIcmNgU



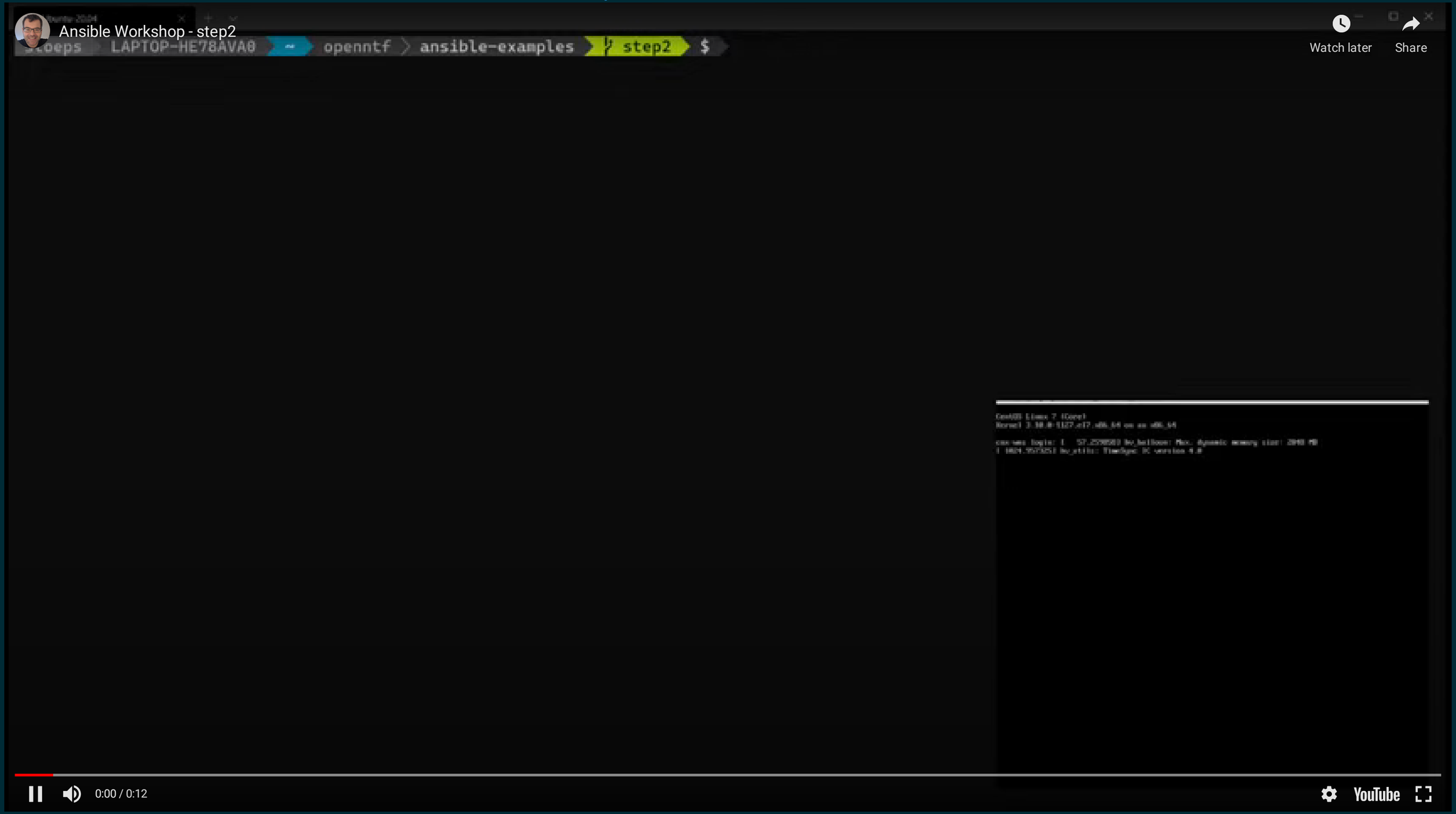
DISPLAY A MESSAGE

playbook.yml

```yaml
---
- hosts: websphere_servers
  become: yes
  become_user: root
  tasks:
    - name: ensure selinux is set to permissive
      selinux:
        policy: targeted
        state: permissive
      register: selinux_status                1

    - debug:
        msg: "SELinux changed. Please reboot the server to apply changes"
      when: selinux_status.changed == true   2
```

1. register a variable to keep the status of this task
2. run only when the task had status changed

youtu.be/HPFuliVmtBE



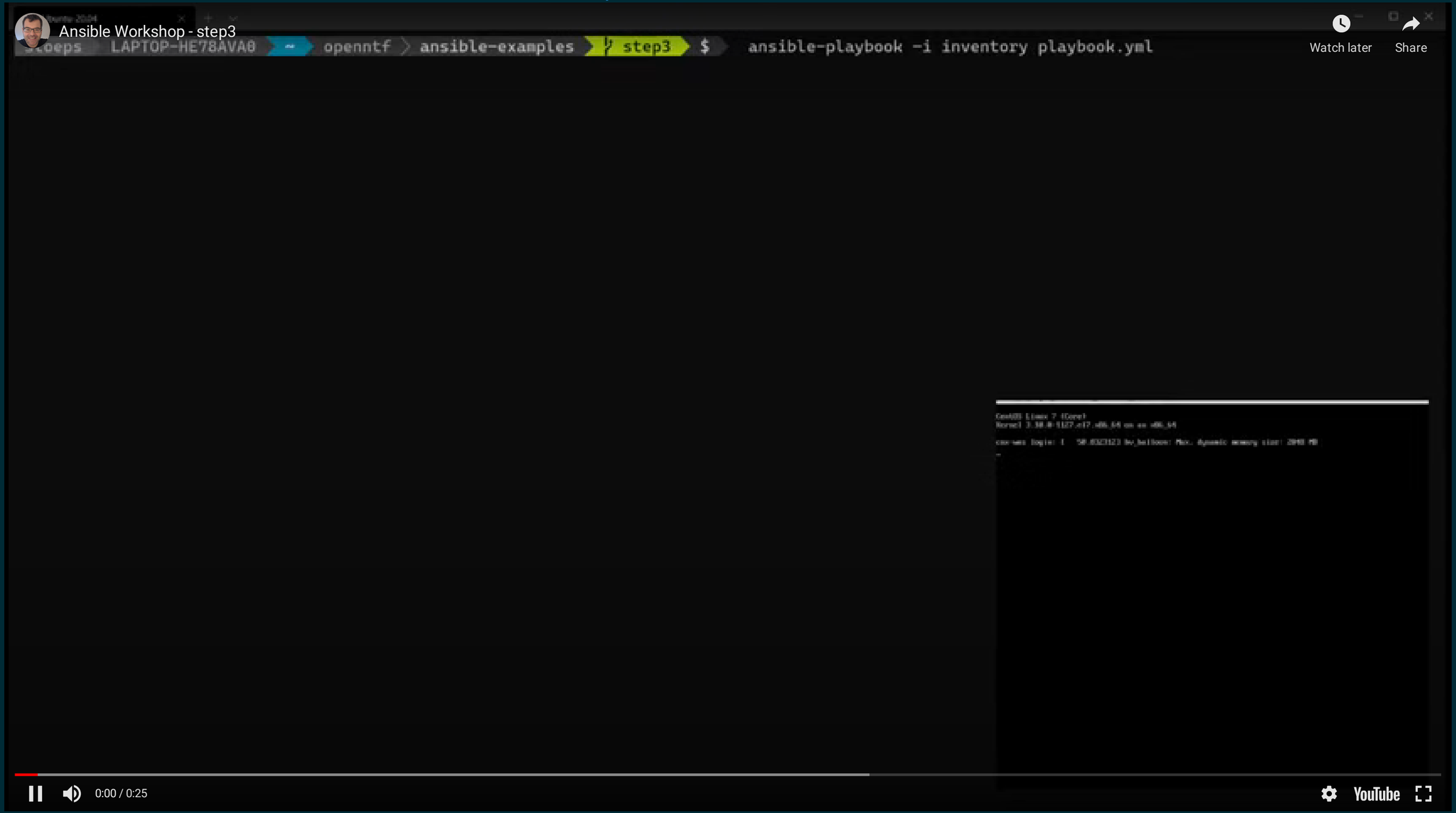
RUN REBOOT AS A TASK

playbook.yml

```yaml
---
- hosts: websphere_servers
  become: yes
  become_user: root
  tasks:
    - name: ensure selinux is set to permissive
      selinux:
        policy: targeted
        state: permissive
      register: selinux_status                    1

    - name: reboot
      reboot:
        msg: "Reboot initiated from Ansible"
        connect_timeout: 30
        reboot_timeout: 120
        test_command: whoami
      when: selinux_status.changed == true
```

1 imagine multiple tasks, you'll end up with tons of variables and complicated when clauses

youtu.be/JeeZMPitUs4



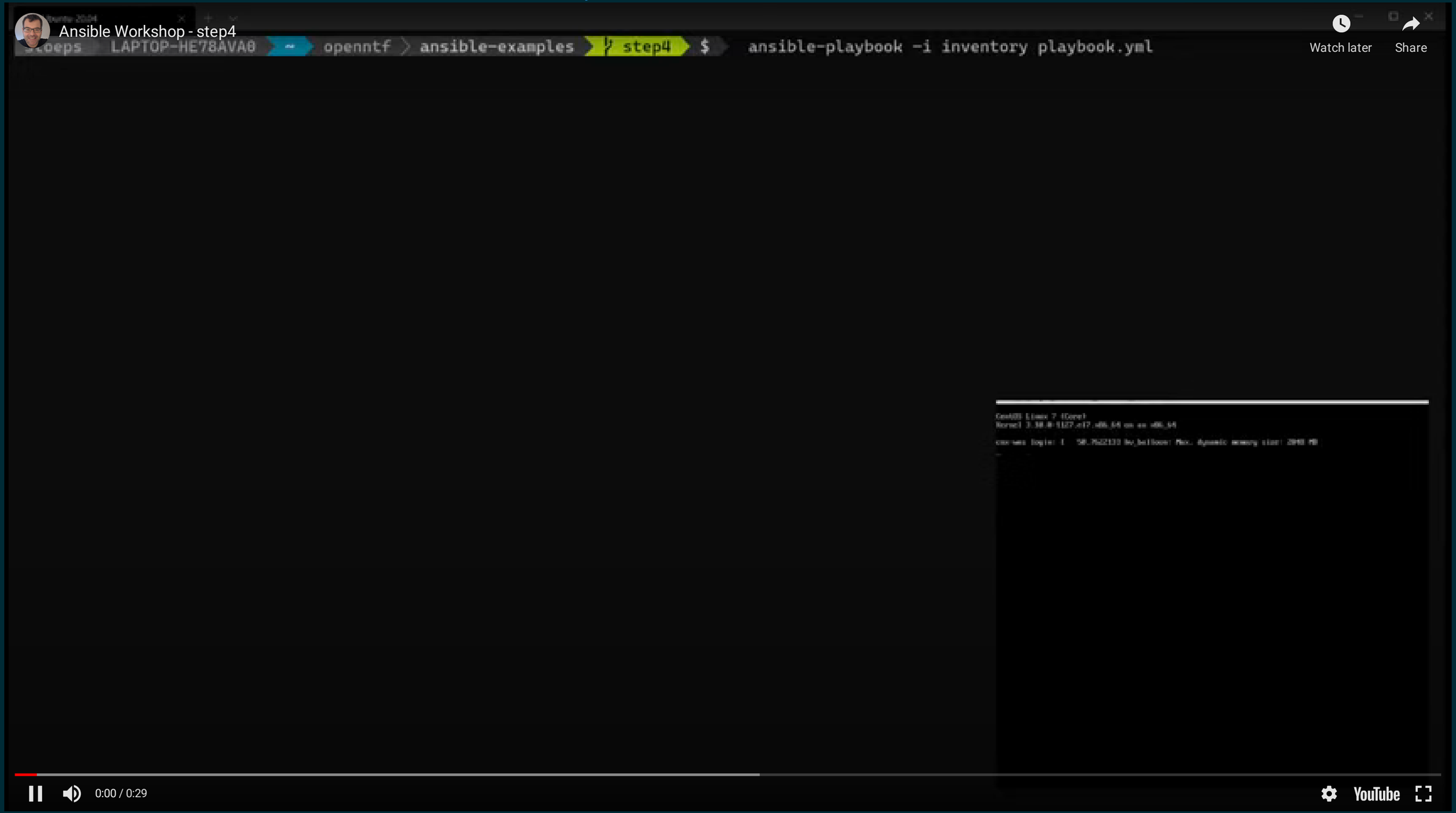
HANDLER

- No need to register a variable
- Just notify the handler (runs only when task status has changed)

```
hosts: websphere_servers
  become: yes
  become_user: root
  tasks:
    - name: ensure selinux is set to permissive
      selinux:
        policy: targeted
        state: permissive
      notify: reboot      1
  handlers:
    - name: reboot
      reboot:
        msg: "Reboot initiated from Ansible"
        connect_timeout: 30
        reboot_timeout: 120
        test_command: whoami
```

1 Notify the handler that status has changed

youtu.be/OLmGwdNncUM

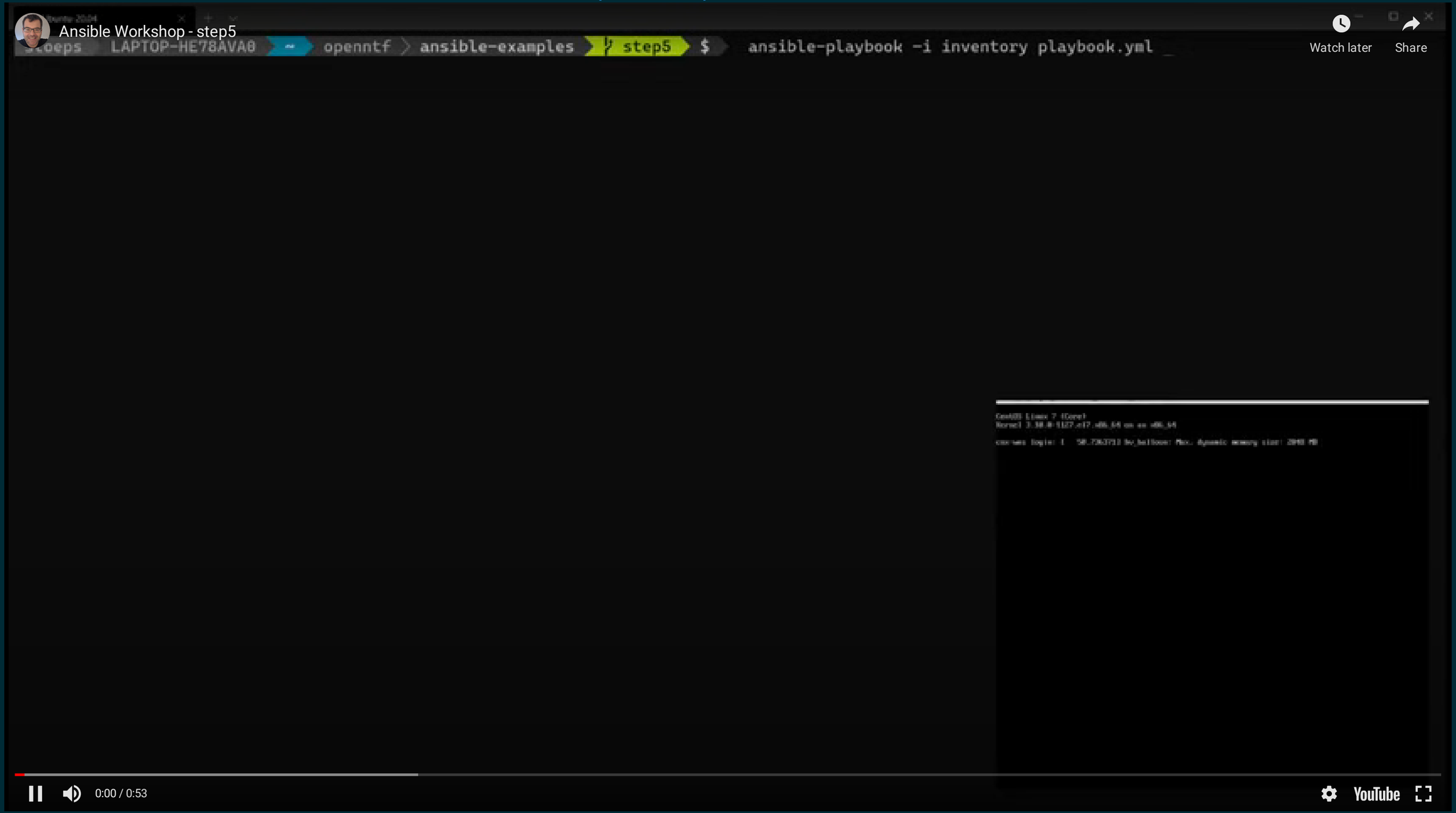


ADD MORE TASKS

```yaml
---
- hosts: websphere_servers
  become: yes
  become_user: root
  tasks:
    - name: ensure selinux is set to permissive
      selinux:
        policy: targeted
        state: permissive
      notify: reboot

    - name: set number of open files in limits.conf
      pam_limits:
        domain: root
        limit_type: '-'
        limit_item: nofile
        value: "65535"
      notify: reboot  1

  handlers:
    - name: reboot
      reboot:
        msg: "Reboot initiated from Ansible"
        connect_timeout: 30
        reboot_timeout: 120
        test_command: whoami
```

1 Reuse the same handler as before (one task must be status changed for a reboot)

youtu.be/ya5TXDRSsHk

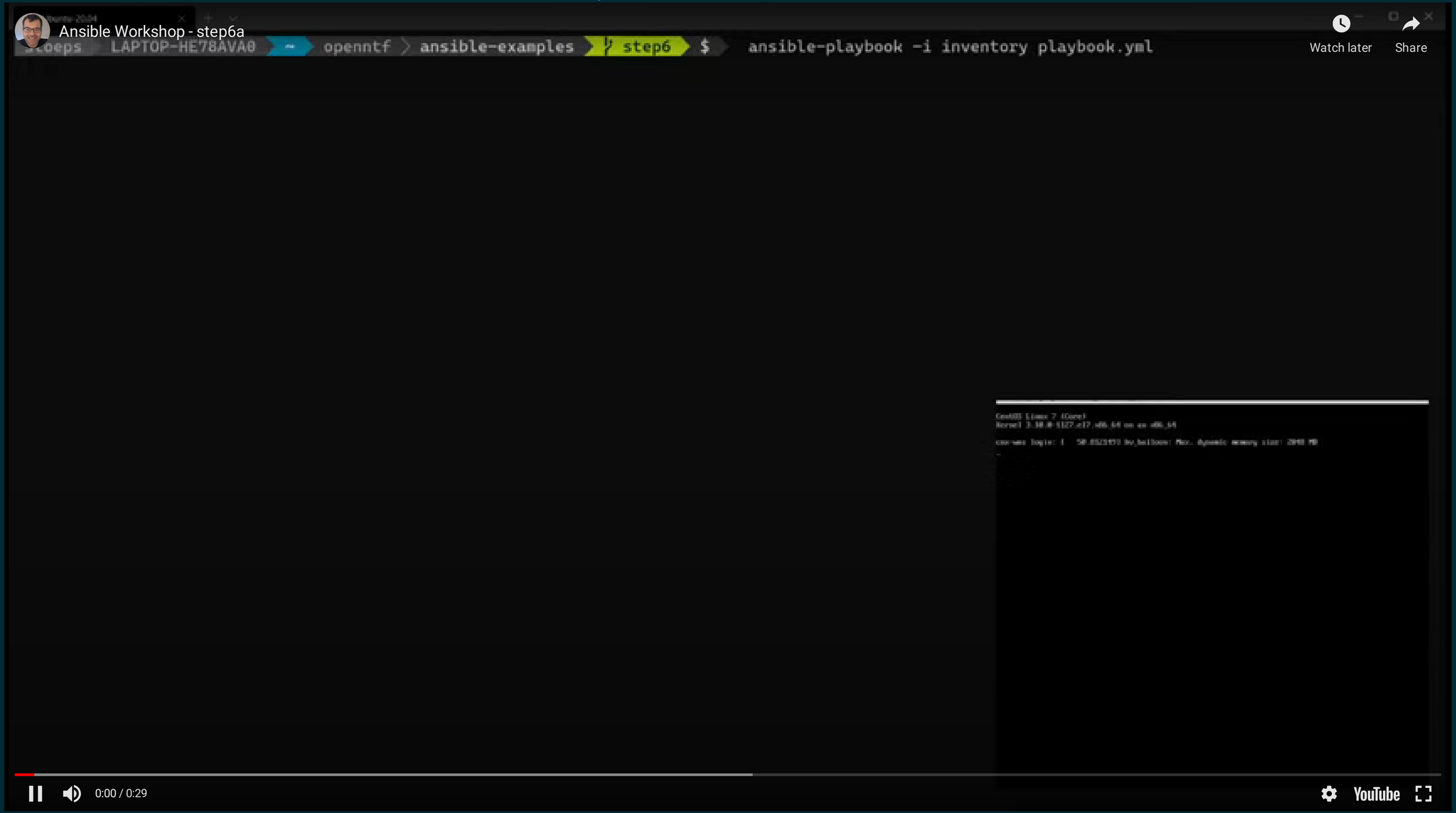


INSTALL A PACKAGE

```yaml
---
- hosts: websphere_servers
  become: yes
  become_user: root
  tasks:
    - name: ensure selinux is set to permissive
      selinux:
        policy: targeted
        state: permissive
      notify: reboot

    - name: Reboot if necessary
      meta: flush_handlers        (1)

    - name: install compatibility package for installation manager
      package:
        name: compat-libstdc++-33.x86_64
        state: present

  handlers:
    - name: reboot
      reboot:
        msg: "Reboot initiated from Ansible"
        connect_timeout: 30
        reboot_timeout: 120
        test_command: whoami
```

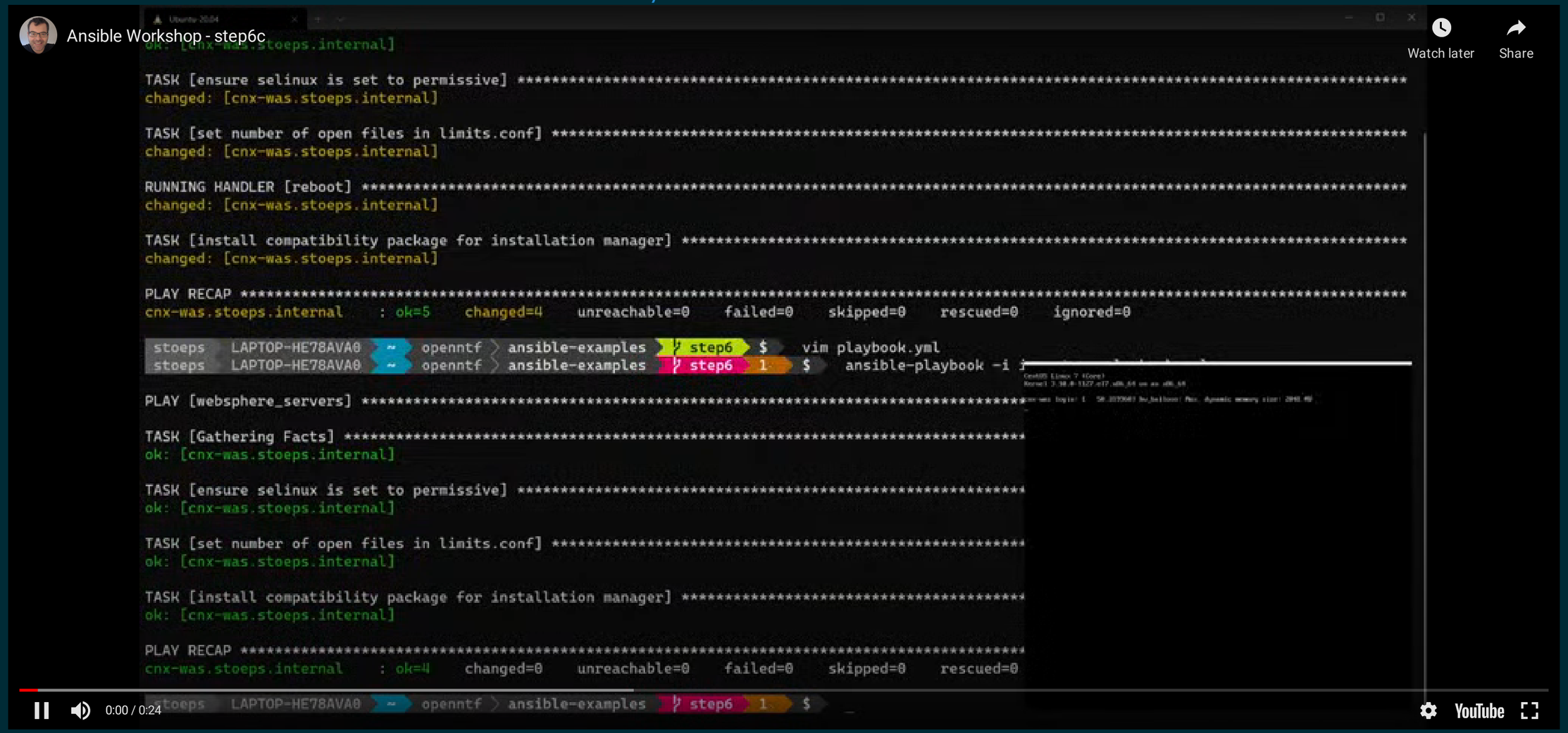
(1) `flush_handler` initiates the handler to run if needed, normally it runs on the end of the role/playbook

youtu.be/HO1dkKIzQd0



REMOVED `flush_handlers`

youtu.be/B4b0LZAhI9c

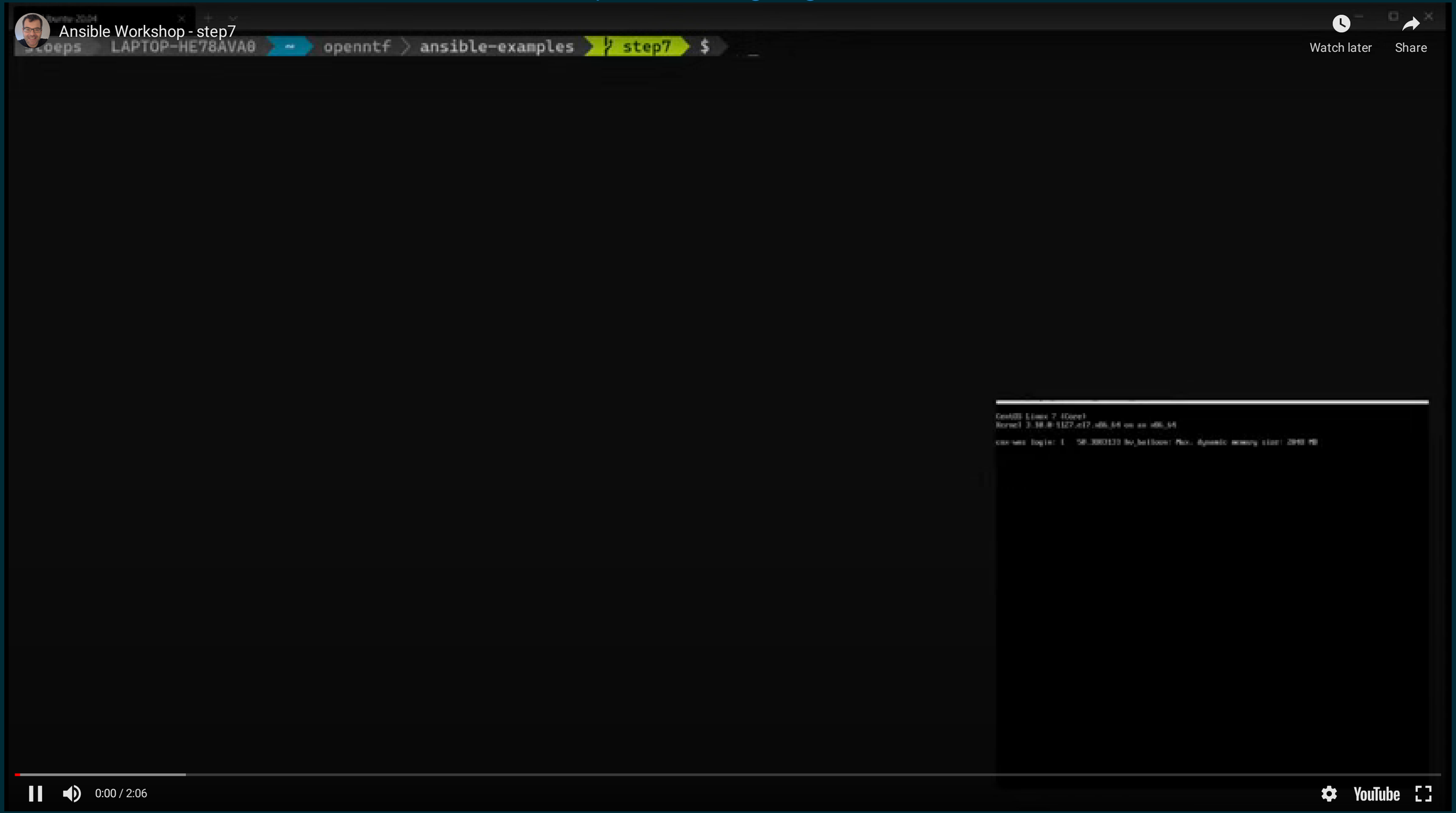


INSTALL MULTIPLE PACKAGES

```
- name: install compatibility packages for installation manager
    package:
      name: "{{ item }}"     1
      state: present
    with_items:              2
      - compat-libstdc++-33.x86_64
      - compat-libstdc++-33.i686
      - libstdc++.x86_64
```

1. placeholder variable
2. all items will be installed

youtu.be/DhGghnYgG0k



ADD ADDITIONAL SERVERS

```
[websphere_servers]
cnx-was.stoeps.internal ansible_host=10.0.11.100

[web_servers]   1
cnx-web.stoeps.internal ansible_host=10.0.11.101

[installationmanager:children]  2
web_servers
websphere_servers
```

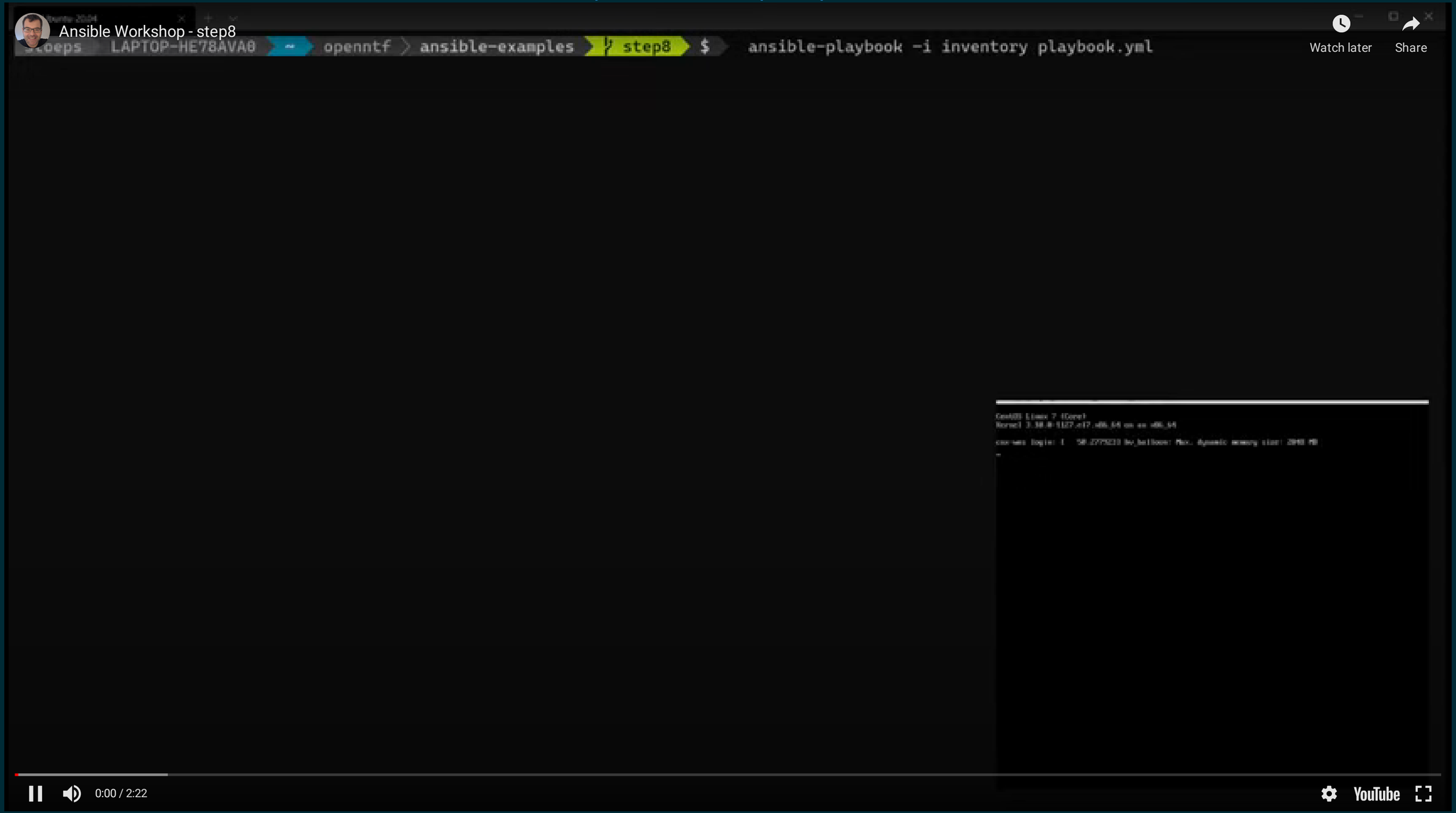
1. Add a second server group
2. Add children of the servergroups to installationmanager

ADD SECOND HOSTGROUP

```
---
- hosts: websphere_servers
  tasks:
    - name: ensure selinux is set to permissive
      selinux:
      [...]

  handlers:
    - name: reboot
      [...]

- hosts: installationmanager    (1)
  tasks:
    - name: install compatibility package for installation manager
      package:
        name: "{{ item }}"
        state: present
      with_items:
        - compat-libstdc++-33.x86_64
        - compat-libstdc++-33.i686
        - libstdc++.x86_64
```

(1) tasks for the new hostgroup (will install package to both server groups

youtu.be/P55Dp5EwpBY

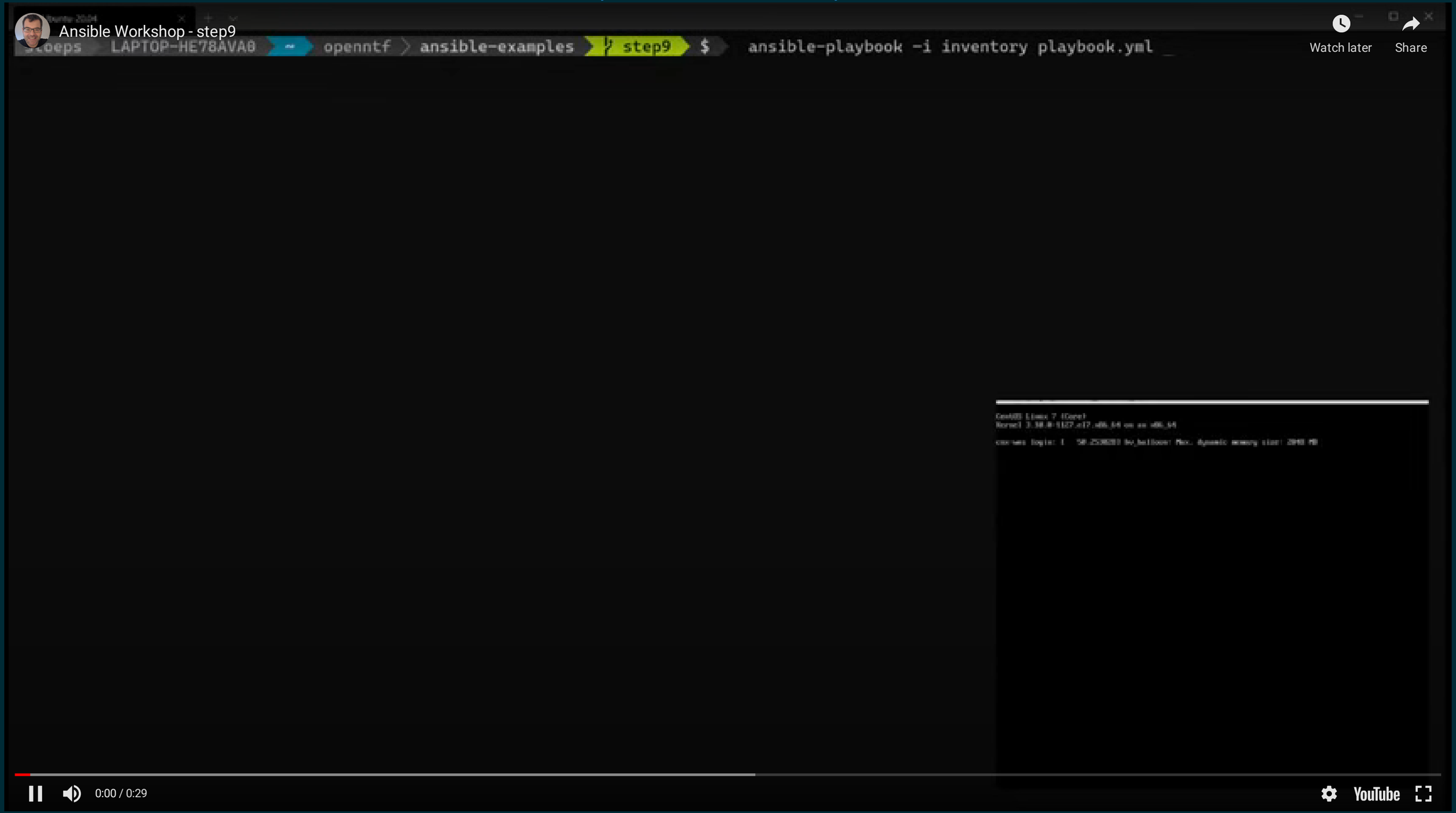


ADD A GROUP AND A USER

```
- name: add group for WebSphere users
  group:
    name: was
    state: present

- name: add user for im and websphere (non_root)
  user:
    name: wassys
    comment: WebSphere user
    uid: 2000
    group: was
    shell: /bin/bash
    state: present
    password: "$6$40GE6/6h6A4UhpBT$kPtpBLe3Komc2bmadagr6S.v0/VRPJoJunEaMl5PBhAb4F5FTWsZff/6CYtTQlVm8Qa2wya4HV
```
1

1 Module needs hash, calculate with `python -c "import crypt; print crypt.crypt('password')"`

youtu.be/z06fB5WRLyE



USE VARIABLES

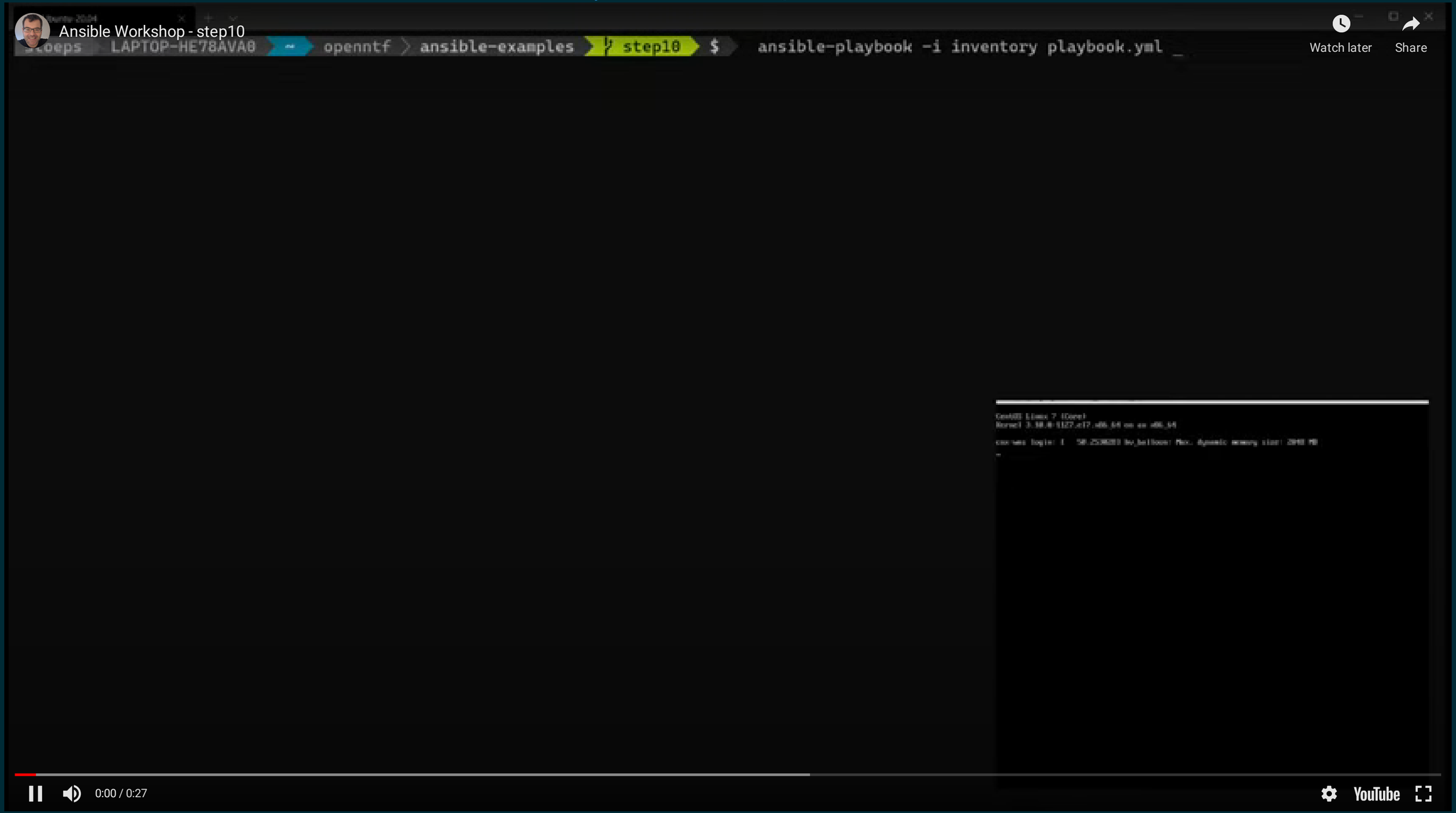
Add to `inventory`

```
...

[installationmanager:vars]
was_user=wassys
was_user_password=password
```

```
- name: hash user password
  shell: "python -c \"import crypt; print crypt.crypt('{{ was_user_password }}')\""  1
  register: was_user_password_hash      2
  changed_when: false
- name: add user for im and websphere (non_root)
  user:
    name: "{{ was_user }}"
    comment: WebSphere user
    uid: 2000
    state: present
    update_password: on_create
    password: "{{ was_user_password_hash.stdout }}"   3
```

1. Calculate the password hash
2. register variable
3. Use stdout (output of hash command) for password hash

youtu.be/GPxHlQuU7N8



CREATE SEPARATE ROLES

playbook.yml

```yaml
- hosts: websphere_servers
  become: yes
  become_user: root
  tasks:      1
    - name: ensure selinux is set to permissive
      selinux:
        policy: targeted
        state: permissive
      notify: reboot
    ...

  handlers:   2
    - name: reboot
      reboot:
        msg: "Reboot initiated from Ansible"
        connect_timeout: 30
        reboot_timeout: 120
        test_command: whoami
```

1. put into `roles/ansible-demo2/tasks/main.yml`
2. put into `roles/ansible-demo2/handlers/main.yml`

VARIABLES DEFAULTS

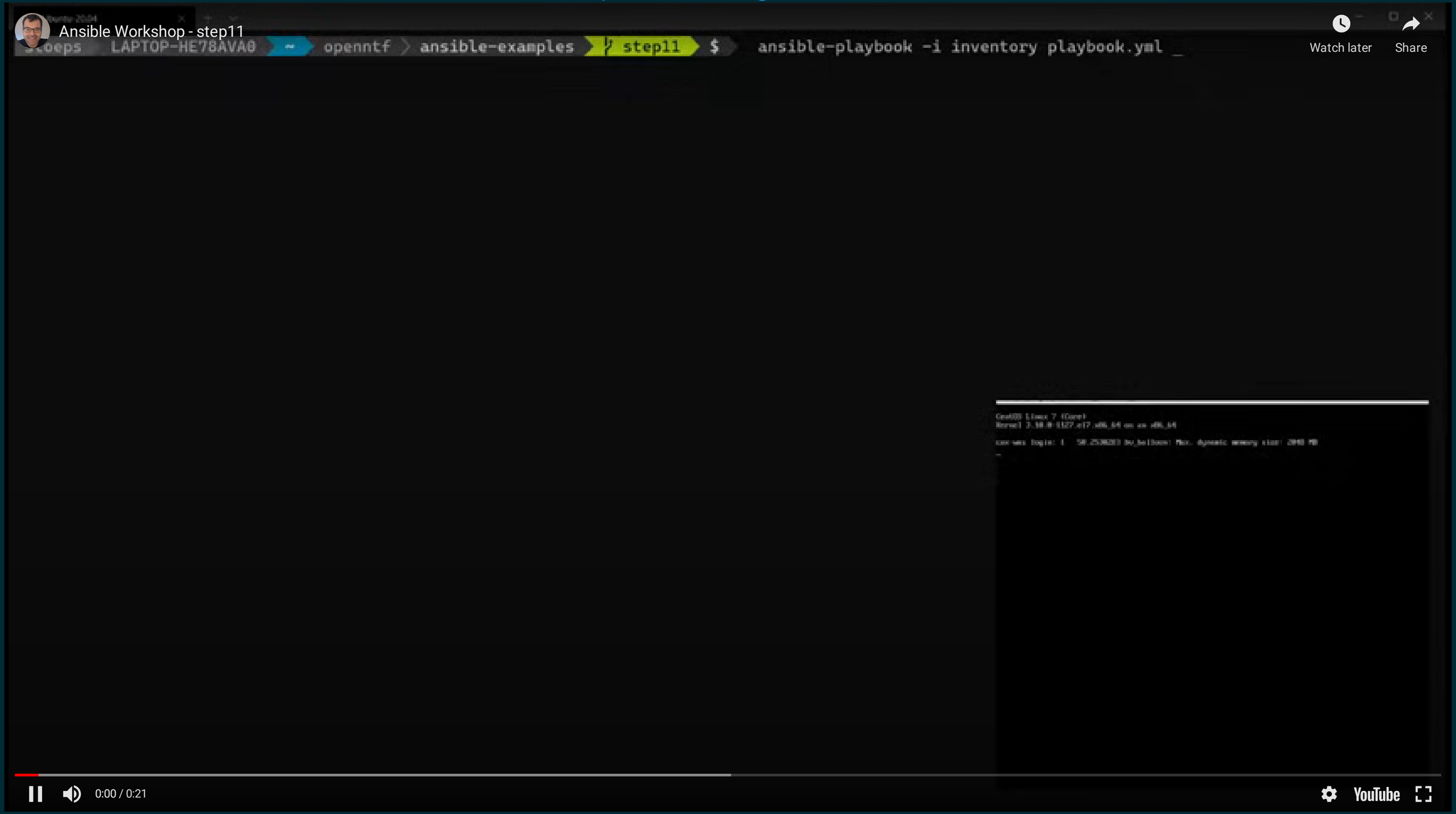
- Add a folder defaults to the role
- Add used variables and their defaults
- So even when you forget to define the variable, the role will run

ansible_demo2/defaults/main.yaml

```
__websphere_user: "{{ was_user | default('wassys') }}"     1
__websphere_user_password: "{{ was_user_password | default('password') }}"   2
```

1. add a variable and read the value from variable was_user, if not present use default wassys
2. default password

youtu.be/Yca0gHKOkxI

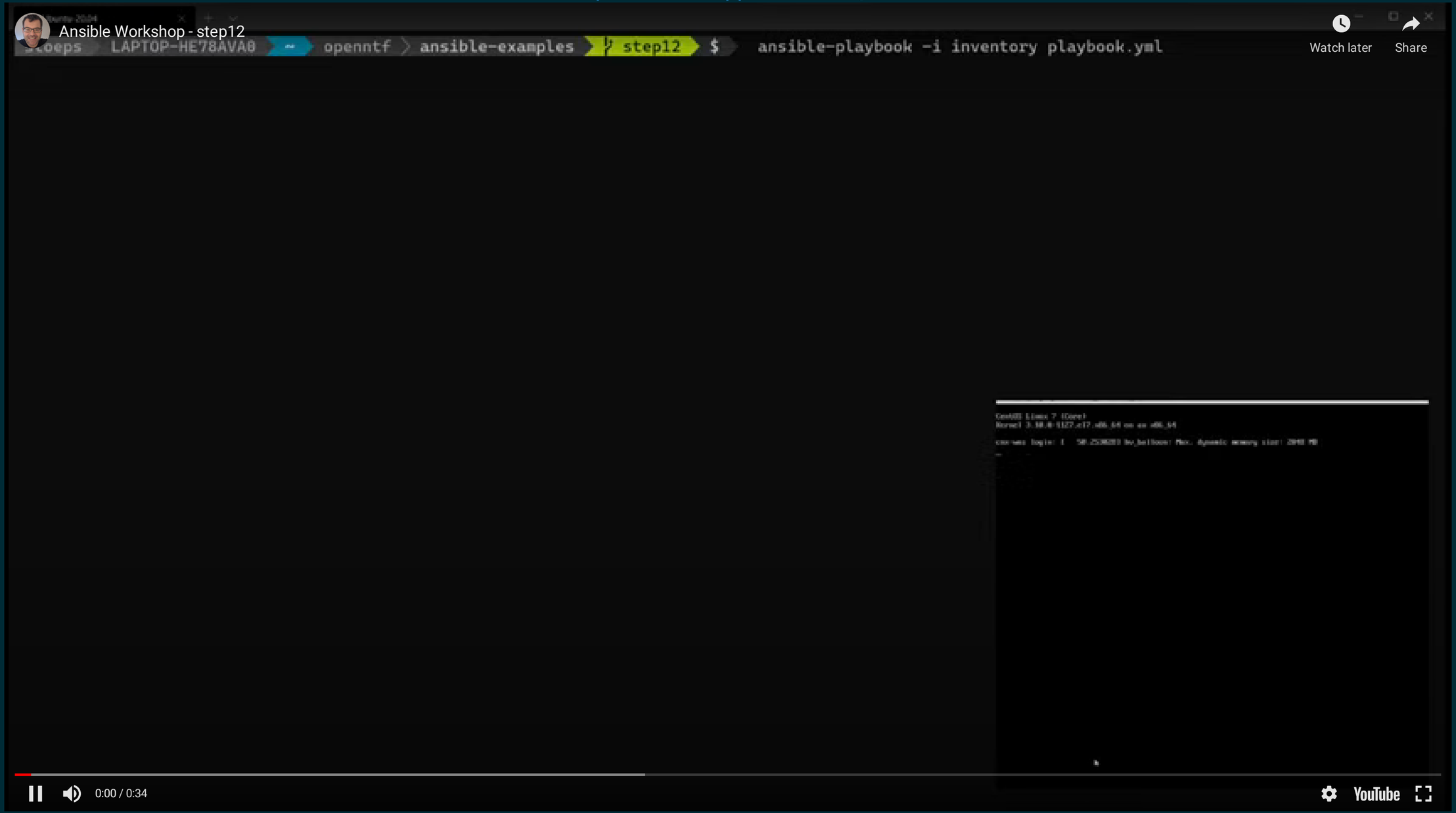


USE ANSIBLE VAULT TO SECURE THE PASSWORD

- move the
  - variables to group_vars/installationmanager.yml
  - passwords to group_vars/all.yaml
- encrypt all.yml

```
ansible-vault encrypt group_vars/all.yml
ansible-playbook -i inventory playbook.yml --ask-vault-pass
```

youtu.be/Ktyy3MKeoRQ



RUN ANSIBLE PLAYBOOK

- Manually through your shell
- Ansible Tower (enterprise server, $$$)
  - On Windows use **Windows Subsystem for Linux** (WSL)
- Ansible AWS (add link)
- Jenkins (Pipeline)
- Directly during provisioning of Vagrant and Terraform

WHERE TO FIND ROLES?

- Simple said: Download or write them
- Check galaxy.ansible.com
- Download role `ansible-playbook install` …
- roles and collections make Ansible modular
- Download complete repositories like connections-automation

SECURITY

- How do we store passwords or deployment keys
- Ansible Vault
  - AES265 encrypted
  - Encrypted during ansible-playbook run
- Ansible AWX
  - Allow users to run tasks and playbooks against hosts without having a root or user account on it

WHERE TO START (LINKS)

- Documentation
  - docs.ansible.com/intro_getting_started.html
  - github.com/orgs/ansible/people
- Books
  - Jeff Geerling: Ansible for Devops
- Youtube
  - Ansible 101 with Jeff Geerling
- Build and deploy container images and containers

ADMINISTRATOR OR DEVELOPER

- Have a look at Ansible
  - Saves you a ton of time
  - Easy to deploy
  - Easy to deploy different environments
    - Dev
    - QA
    - Test
    - Production
- KISS

CONNECTIONS CUSTOMERS

- Have a look at
  - github.com/HCL-TECH-SOFTWARE/connections-automation