



20th International Conference on Network and Service Management

Prague, Czech Republic // 28 - 31 October 2024

AI-Powered Network and Service Management for Tomorrow's Digital World



How to Ansible

Richard Plný
FIT CTU in Prague
plnyrich@fit.cvut.cz

27th October 2024
Ph.D. School, Prague
Czech Republic

Outline

1. Basics
2. Installation
3. Ansible
4. Demo

Part 1 - Basics

Problem

- We often need to configure a new server quickly..
 - Deployment of an application
 - Prepare an exact environment for research
 - Manage cloud machines
- A lot of work manually...

=> **Ansible**

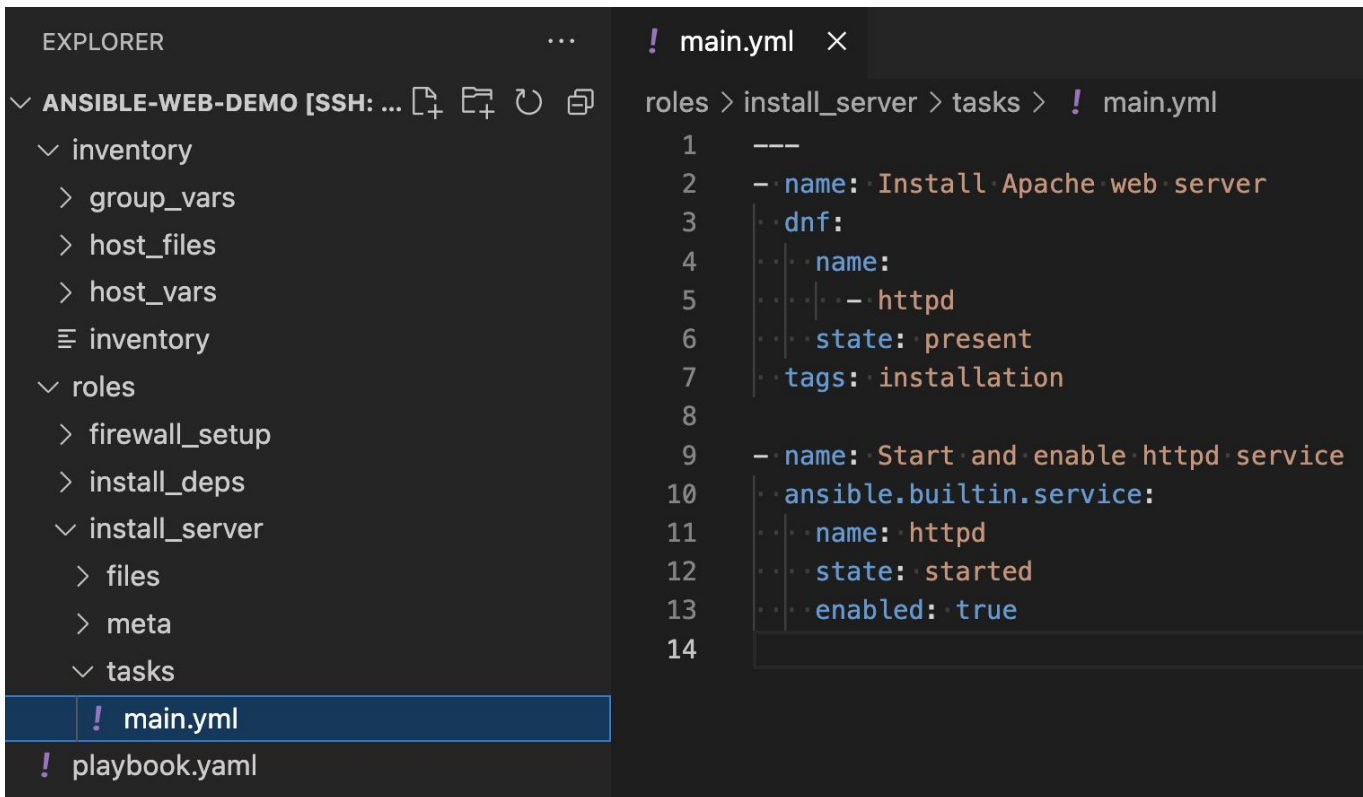


What is Ansible?

- Automation engine for:
 - Provisioning
 - Configuration Management
 - Application Deployment
 - Orchestration
- YAML format
- Main website: <https://www.ansible.com>
- GitHub: <https://github.com/ansible/ansible>



Example



The image shows a screenshot of a code editor with a dark theme. On the left is the 'EXPLORER' sidebar showing a project named 'ANSIBLE-WEB-DEMO'. The project structure includes folders for 'inventory', 'roles', and 'tasks'. The 'roles' folder is expanded, showing sub-folders like 'firewall_setup', 'install_deps', and 'install_server'. The 'install_server' folder is also expanded, showing 'files', 'meta', and 'tasks'. The 'tasks' folder is selected, and the file 'main.yml' is highlighted. On the right, the content of 'main.yml' is displayed. It shows a task named 'Install Apache web server' using the 'dnf' module to install 'httpd'. The task is part of a role named 'install_server' and is tagged for 'installation'. The task is also configured to start and enable the 'httpd' service.

```
EXPLORER
ANSIBLE-WEB-DEMO [SSH: ...]
├── inventory
│   ├── group_vars
│   ├── host_files
│   ├── host_vars
│   └── inventory
├── roles
│   ├── firewall_setup
│   ├── install_deps
│   └── install_server
│       ├── files
│       ├── meta
│       └── tasks
└── main.yml
playbook.yaml
```

```
! main.yml x
roles > install_server > tasks > ! main.yml
1 ---
2 - name: Install Apache web server
3   dnf:
4     name:
5     - httpd
6     state: present
7     tags: installation
8
9 - name: Start and enable httpd service
10  ansible.builtin.service:
11    name: httpd
12    state: started
13    enabled: true
14
```

Where is it used?

1. GitHub repositories using Docker/Vagrant to prepare the machine
2. Gitlab CI
3. App deployment
4. Cloud provisioning

```
main.yaml 657 B
1 ---
2 - name: Python {{ version }} is installed
3   ansible.builtin.dnf:
4     name:
5       - "{{ python }}"
6       - "{{ python }}-pip"
7       - "{{ python }}-setuptools"
8       - "{{ python }}-devel"
9       - "{{ python }}-numpy"
10    state: latest
11  vars:
12    python: "python{{ version_major }}{{ version_minor }}"
13  when:
14    - ansible_os_family == 'RedHat'
15
16 - name: Set Python {{ version }} as default interpreter
17   ansible.builtin.command: "alternatives --set {{ python }} {{ bin }}"
18  vars:
19    python: "python{{ version_major }}"
20    bin: "/usr/bin/python{{ version_major }}.{{ version_minor }}"
21  when:
22    - ansible_os_family == 'RedHat'
23    - set_as_default
24
```

Alternatives

- Attune
 - <https://attuneops.io>
- Puppet
 - <https://www.puppet.com>
- Terraform
 - <https://www.terraform.io>
- Chef
 - <https://www.chef.io>



AttuneOps



puppet



Progress[®] Chef[®]

Part 2 - Installation

Installation

- Minimal installation

```
pip3 install ansible-core
```

- Larger package with additional community-selected roles

```
pip3 install --include-deps ansible
```

- Also possible with dnf/yum

```
sudo {dnf,yum} install ansible
```

- Python3 on the managed machines

- https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

Ansible Galaxy

- Ansible's "pip"
`ansible-galaxy role install namespace.role_name`
- `requirements.yml`
- Database : <https://galaxy.ansible.com/>
- Docs: https://docs.ansible.com/ansible/latest/galaxy/user_guide.html

Ansible Galaxy

The screenshot displays the Ansible Galaxy web interface. On the left is a dark sidebar with navigation links: 'Ansible Galaxy', 'Search', 'Collections' (with a dropdown arrow), 'Namespaces', 'Roles' (with a right arrow), 'Documentation' (with an external link icon), and 'Terms of Use' (with an external link icon). The 'Collections' dropdown is open, showing 'Collections' and 'Namespaces'. The main content area has a dark header with the 'A GALAXY' logo, 'English' language selector, and a 'Login' link. Below the header is a message: 'Thanks for trying out the new and improved Galaxy, please share your feedback on forum.ansible.com'. The breadcrumb trail is 'Namespaces > cisco > catalystwan'. The collection name 'cisco.catalystwan' is prominently displayed with the Cisco logo. Below it, the version '0.2.2 updated 16 days ago (latest)' is shown, along with a 'Last updated 16 days ago' note and '465 Downloads'. A horizontal menu contains 'Install' (active), 'Documentation', 'Contents', 'Import log', and 'Dependencies'. On the right of this menu are links for 'Docs site', 'Website', 'Issue tracker', and 'Repo'. The 'Install' section is titled 'Install' and describes the collection as 'Ansible collection to interact with Cisco SD-WAN vManage via catalystwan'. It features tags for 'catalystwan', 'sdwan', 'cisco', and 'networking'. The license is 'GPL-3.0-only'. The installation command is shown in a box: 'ansible-galaxy collection install cisco.catalystwan'. A note states: 'Note: Installing collections with ansible-galaxy is only supported in ansible-core>=2.13.9'. The download section shows a 'Download tarball' link. At the bottom, it specifies 'Requires Ansible >=2.16.6'.

Ansible Galaxy

English ? Login

Thanks for trying out the new and improved Galaxy, please share your feedback on forum.ansible.com.

Namespaces > cisco > catalystwan

cisco.catalystwan

Version 0.2.2 updated 16 days ago (latest) Last updated 16 days ago 465 Downloads

Install Documentation Contents Import log Dependencies Docs site Website Issue tracker Repo

Install

Ansible collection to interact with Cisco SD-WAN vManage via catalystwan

catalystwan sdwan cisco networking

License GPL-3.0-only

Installation `ansible-galaxy collection install cisco.catalystwan`

Note: Installing collections with ansible-galaxy is only supported in ansible-core>=2.13.9

Download [Download tarball](#)

Requires Ansible >=2.16.6

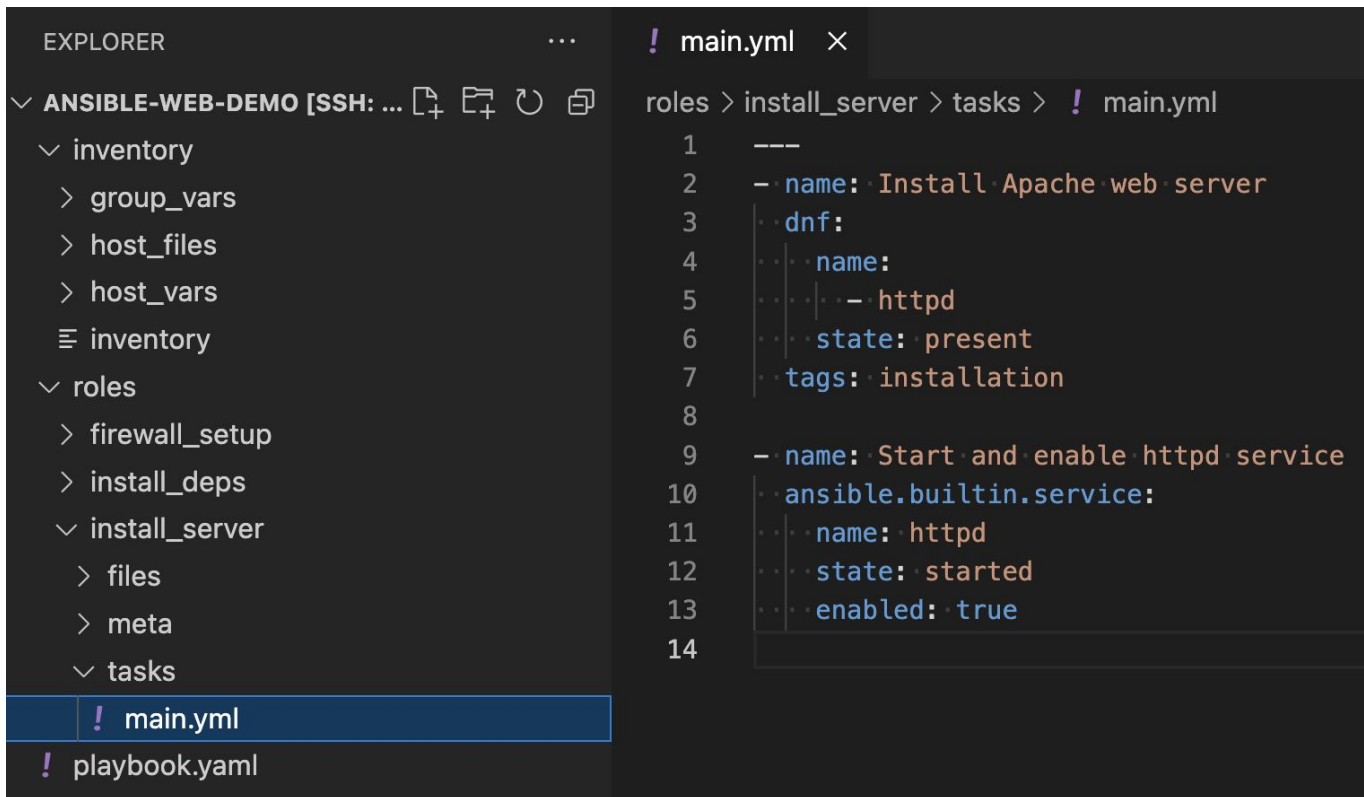
Part 3 - Ansible

Repository with sources

<https://github.com/plnyrich/Ansible-Demo>

```
vagrant up --provider=virtualbox
```

Structure



EXPLORER

- ANSIBLE-WEB-DEMO [SSH: ...]
 - inventory
 - group_vars
 - host_files
 - host_vars
 - inventory
 - roles
 - firewall_setup
 - install_deps
 - install_server
 - files
 - meta
 - tasks
 - ! main.yml
 - ! playbook.yml

roles > install_server > tasks > ! main.yml

```
1 ---
2 - name: Install Apache web server
3   dnf:
4     name:
5       - httpd
6     state: present
7     tags: installation
8
9 - name: Start and enable httpd service
10  ansible.builtin.service:
11    name: httpd
12    state: started
13    enabled: true
14
```

Inventory

- YAML or .ini format
- Define servers and their roles
 - Login credentials can be defined
 - `ansible_ssh_user`
 - `ansible_ssh_pass`
 - Ranges (inclusive)
 - `db_server_[0:2].company.com =>`
`db_server_0.company.com`
`db_server_1.company.com`
`db_server_2.company.com`

```
inventory > [web_servers]
1 web_server1.fit.cvut.cz
2 web_server2.fit.cvut.cz
3 #web_server[1:2].fit.cvut.cz
4
5 [db_servers]
6 db_server1.fit.cvut.cz ansible_user=dbmanager ansible_become=yes ansible_become_method=sudo
7
8 [local]
9 localhost
10
```

- https://docs.ansible.com/ansible/latest/inventory_guide/intro_inventory.html

Inventory

! inventory.yaml ✕

ansible-demo > inventory > ! inventory.yaml

```
1  web_servers:
2  |  · hosts:
3  |  |  · web_server1.fit.cvut.cz
4  |  |  · web_server2.fit.cvut.cz
5
6  db_servers:
7  |  · hosts:
8  |  |  · db_server1.fit.cvut.cz:
9  |  |  |  · ansible_user: dbmanager
10 |  |  |  · ansible_become: yes
11 |  |  |  · ansible_become_method: sudo
12
13 local:
14 |  · hosts:
15 |  |  · localhost
```

≡ inventory ✕

inventory > ≡ inventory

```
1  [web_servers]
2  web_server1.fit.cvut.cz
3  web_server2.fit.cvut.cz
4  #web_server[1:2].fit.cvut.cz
5
6  [db_servers]
7  db_server1.fit.cvut.cz ansible_user=dbmanager ansible_become=yes ansible_become_method=sudo
8
9  [local]
10 localhost
```

Playbooks

- Define what should be done on what machines
- “Roles” are executed in the defined order
- `--check` for syntax check without running the commands

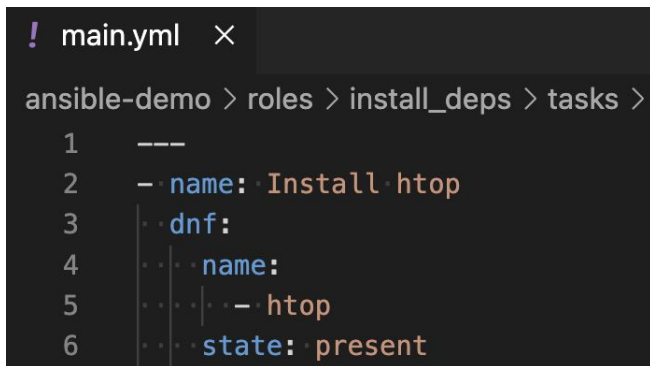
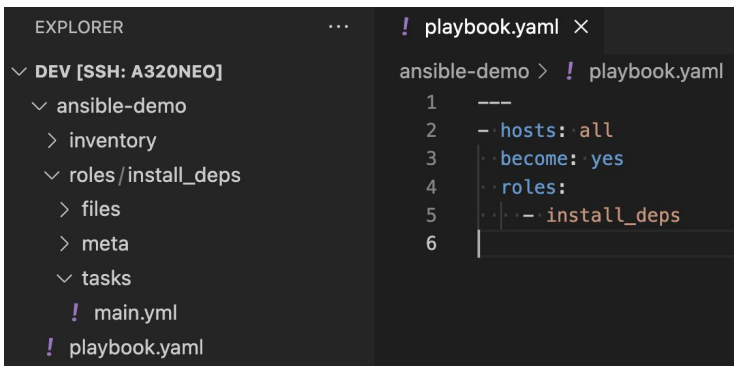
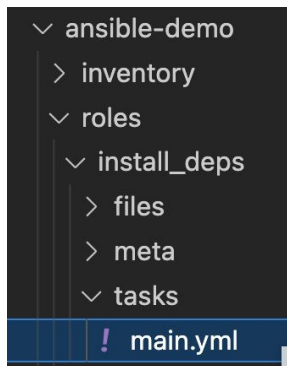
`ansible-playbook -i inventory playbook.yml`

- https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_intro.html

```
! playbook.yml ×
ansible-demo > ! playbook.yml
1  ---
2  - hosts: web_servers
3    become: yes
4    roles:
5      - install_deps
6      - install_web_server
7      - firewall_setup
8
9  - hosts: db_servers
10    become: yes
11    roles:
12      - install_deps
13      - install_db_server
14      - setup_backups
15      - firewall_setup
16
```

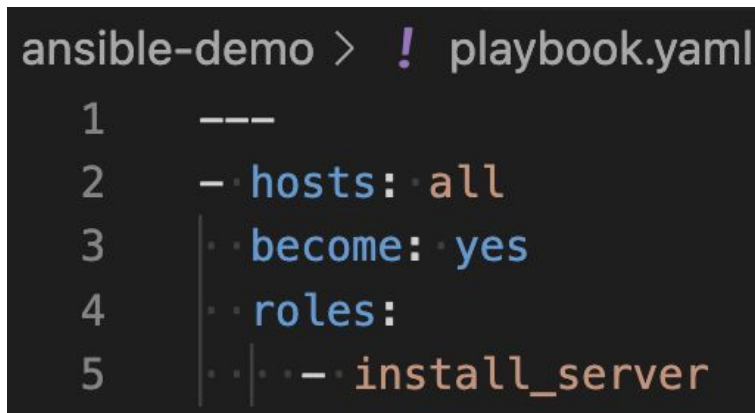
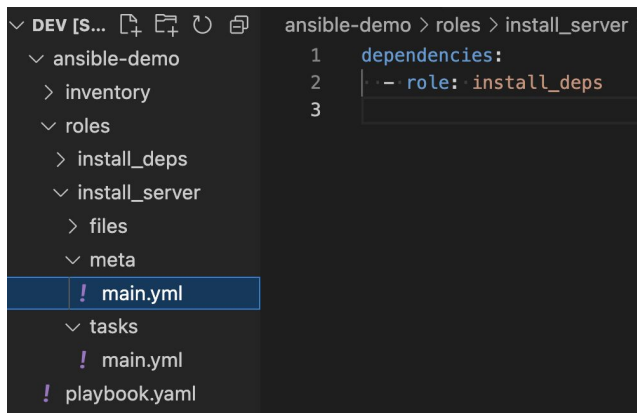
Roles

- Playbooks composed from roles
- Must be in **roles/** directory
- Name of folder in roles/ can be used in playbook
- Role is set of tasks, task is one action



Role Dependencies

- We do not want to install web server before we install dependencies
=> **meta/main.yml**
- https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_intro.html



Role Dependencies

```
● [plnyrich@a320neo ansible-demo]$ ansible-playbook -i inventory/inventory_localhost playbook.yaml

PLAY [all] *****

TASK [Gathering Facts] *****
[WARNING]: Platform linux on host localhost is using the discovered Python interpreter at /usr/bin/python3.9,
but future installation of another Python interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-core/2.15/reference\_appendices/interpreter\_discovery.html for more
information.
ok: [localhost]

TASK [install_deps : Install http] *****
ok: [localhost]

TASK [install_server : Install web server] *****
changed: [localhost]

PLAY RECAP *****
localhost                : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Variables

- Sometimes we need to customize what should be done
=> **variables**
- Use variable in role: `{{ varName }}`
- Some vars are prepared by Ansible
 - `ansible_facts`
 - `inventory_dir`
 - ...
- https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html
- https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_vars_facts.html

```
! playbook.yaml ×
ansible-demo > ! playbook.yaml
1  ---
2  - hosts: web_servers
3    become: yes
4    vars:
5      httpPort: 8080
6      filePaths:
7        - path1
8        - path2
9        - path3
10   roles:
11     - install_deps
12     - install_web_server
13     - firewall_setup
14
15  - hosts: db_servers
16    become: yes
17    vars:
18      backupDest: "/var/backup"
19    roles:
20      - install_deps
21      - install_db_server
22      - setup_backups
23      - firewall_setup
```

Variables

ansible-demo > roles > env_prep > tasks > ! main.yml

```
1    ---
2    - name: Ensure file is deleted
3      file:
4        path: "{{ filePaths[0] }}"
5        state: absent
6
7    - name: Ensure port is open
8      become: yes
9      firewallld:
10        port: "{{ httpPort }}/tcp"
11        permanent: true
12        immediate: true
13        state: enabled
```

Loops

- We want to install N packages, delete X files, ...
- `loop, with_items`
 - `with_items`: single-level flattening
 - `loop: "{{ ... | flatten(1) }}"` is equivalent to `with_items`
- https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_loops.html

```
ansible-demo > roles > install_deps
1  ---
2  - name: Install deps
3    dnf:
4      name:
5        - "{{ item }}"
6      state: present
7    loop:
8      - htop
9      - wget
```

```
ansible-demo > roles > env_prep > tasks >
1  ---
2  - name: Ensure file is deleted
3    file:
4      path: "{{ item }}"
5      state: absent
6    loop: "{{ filePaths }}"
```

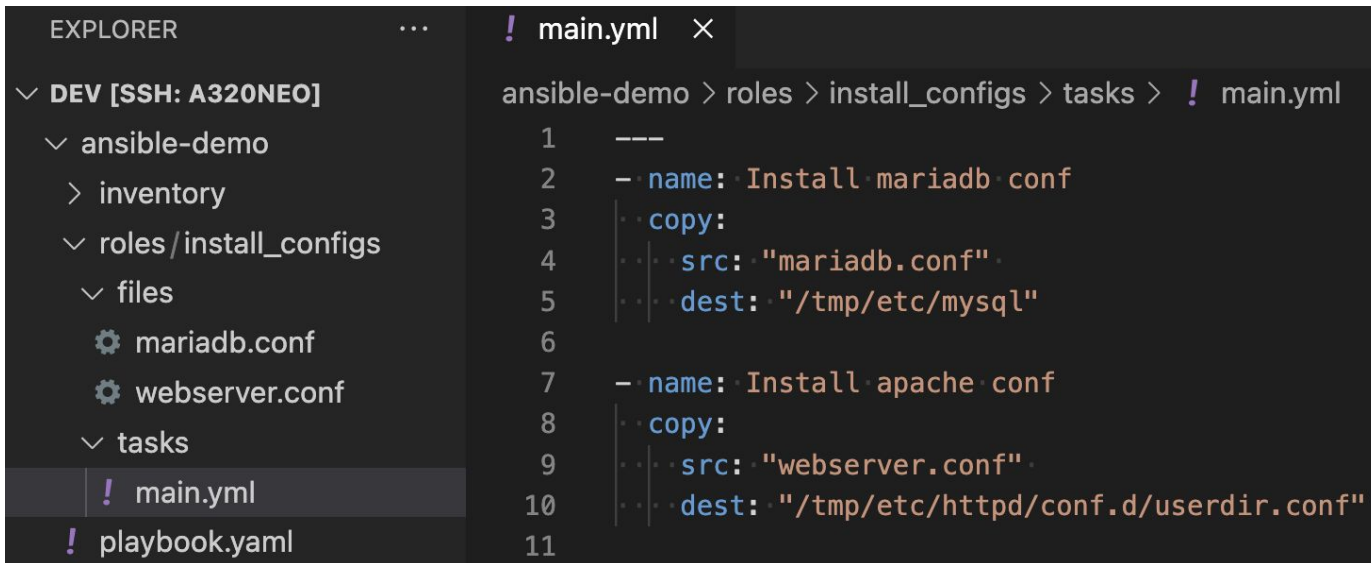

Loops

- We can also define more complex items or “objects”

```
ansible-demo > roles > confs > tasks > ! main.yml
1  ---
2  - name: Create config files
3    copy:
4      dest: "/tmp/{{ item.fileName }}"
5      content: "{{ item.content }}"
6    loop:
7      - { fileName: 'file1', content: "l1\nl2\nl3\n" }
8      - { fileName: 'file2', content: "l4\nl5\nl6\n" }
```

Files

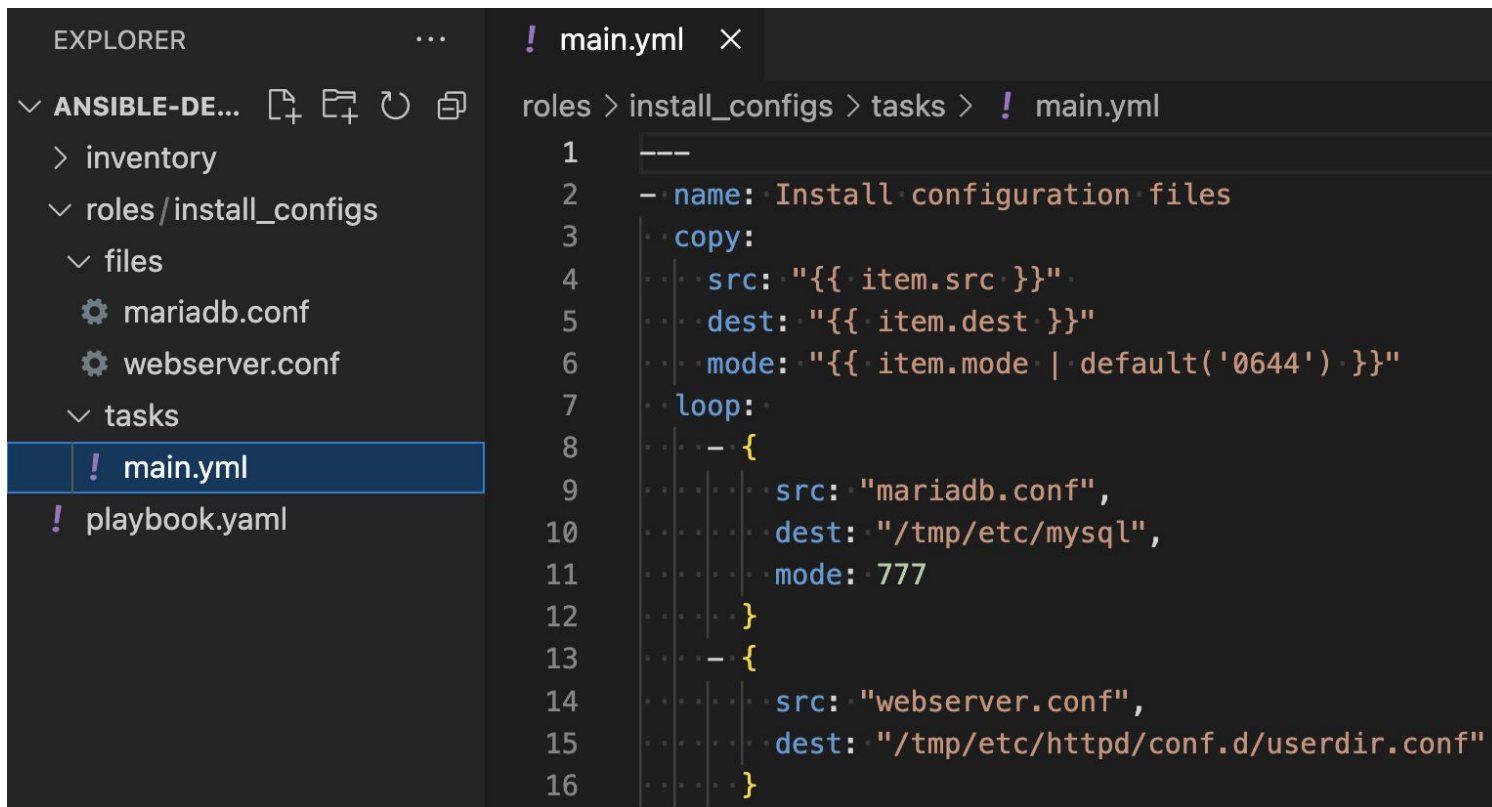
- Prepare configuration files, ...
- https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_reuse_roles.html



The screenshot shows an IDE interface with a dark theme. On the left is the 'EXPLORER' sidebar showing a project structure for 'DEV [SSH: A320NEO]'. The structure includes a directory 'ansible-demo' containing 'inventory', a role 'roles/install_configs' containing 'files' (with 'mariadb.conf' and 'webserver.conf') and 'tasks' (with 'main.yml' and 'playbook.yaml'). The 'main.yml' file is selected. The main editor area shows the content of 'main.yml' with a breadcrumb path: 'ansible-demo > roles > install_configs > tasks > ! main.yml'. The file content is as follows:

```
1 ---
2 - name: Install mariadb.conf
3   copy:
4     src: "mariadb.conf"
5     dest: "/tmp/etc/mysql"
6
7 - name: Install apache.conf
8   copy:
9     src: "webserver.conf"
10    dest: "/tmp/etc/httpd/conf.d/userdir.conf"
11
```

Files

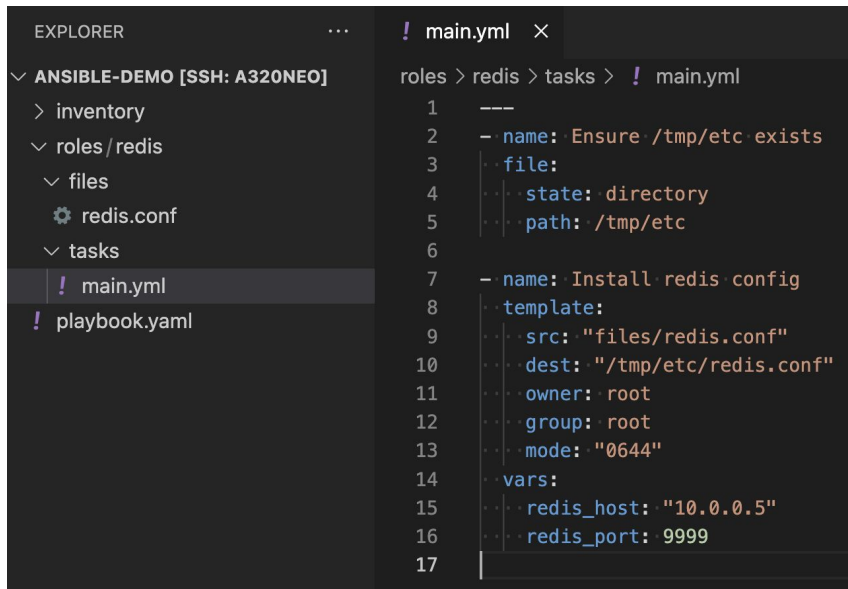


```
EXPLORER
ANSIBLE-DE...
  > inventory
  > roles/install_configs
    > files
      mariadb.conf
      webserver.conf
    > tasks
      ! main.yml
      ! playbook.yml

! main.yml x
roles > install_configs > tasks > ! main.yml
1 ---
2 - name: Install configuration files
3   copy:
4     src: "{{ item.src }}"
5     dest: "{{ item.dest }}"
6     mode: "{{ item.mode | default('0644') }}"
7   loop:
8     - {
9         src: "mariadb.conf",
10        dest: "/tmp/etc/mysql",
11        mode: 777
12      }
13    - {
14        src: "webserver.conf",
15        dest: "/tmp/etc/httpd/conf.d/userdir.conf"
16      }
```

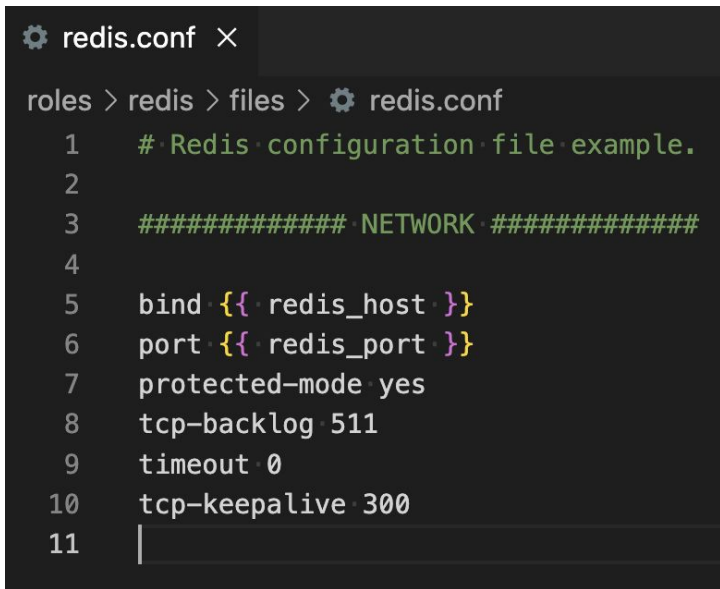
Templates

- Use vars in the prepared files in **files/**
- https://docs.ansible.com/ansible/latest/collections/ansible/builtin/template_module.html



The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays a project named 'ANSIBLE-DEMO [SSH: A320NEO]' with a file tree containing 'inventory', 'roles/redis' (with sub-items 'files' and 'redis.conf'), and 'tasks' (with sub-items 'main.yml' and 'playbook.yaml'). The 'main.yml' file is selected and open in the editor. The breadcrumb navigation at the top of the editor reads 'roles > redis > tasks > ! main.yml'. The code in the editor is as follows:

```
1 ---
2 - name: Ensure /tmp/etc exists
3   file:
4     state: directory
5     path: /tmp/etc
6
7 - name: Install redis config
8   template:
9     src: "files/redis.conf"
10    dest: "/tmp/etc/redis.conf"
11    owner: root
12    group: root
13    mode: "0644"
14  vars:
15    redis_host: "10.0.0.5"
16    redis_port: 9999
17
```



The screenshot shows the VS Code editor with the 'redis.conf' file open. The breadcrumb navigation at the top of the editor reads 'roles > redis > files > ! redis.conf'. The code in the editor is as follows:

```
1 # Redis configuration file example.
2
3 ##### NETWORK #####
4
5 bind {{ redis_host }}
6 port {{ redis_port }}
7 protected-mode yes
8 tcp-backlog 511
9 timeout 0
10 tcp-keepalive 300
11
```

Templates

```
● [plnyrich@a320neo ansible-demo]$ ansible-playbook -i inventory/inventory_localhost playbook.yaml

PLAY [all] *************************************************************************************************************************************

TASK [Gathering Facts] *************************************************************************************************************************************
[WARNING]: Platform linux on host localhost is using the discovered Python interpreter at /usr/bin/python3.9, but future
installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.15/reference_appendices/interpreter_discovery.html for more information.
ok: [localhost]

TASK [redis : Ensure /tmp/etc exists] *********************************************************************
ok: [localhost]

TASK [redis : Install redis config] *********************************************************************
changed: [localhost]

PLAY RECAP *********************************************************************
localhost                : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

● [plnyrich@a320neo ansible-demo]$ cat /tmp/etc/redis.conf
# Redis configuration file example.

##### NETWORK #####

bind 10.0.0.5
port 9999
protected-mode yes
tcp-backlog 511
timeout 0
tcp-keepalive 300
○ [plnyrich@a320neo ansible-demo]$
```

Tags

- Mark roles with tags and run only some of them
- Example: installation, configuration, ...
- runs only roles marked with **hello** tag:

```
ansible-playbook -i inventory/inventory_localhost playbook.yml --tags 'hello'
```

- Runs everything except **hi** tag

```
ansible-playbook -i inventory/inventory_localhost playbook.yml --skip-tags 'hi'
```

- https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_tags.html

```
ansible-demo > roles > say_hello
1  ---
2  - name: Say hello
3    debug:
4      msg: Hello
5      tags: hello
6
7  - name: Say hi
8    debug:
9      msg: Hi
10   tags: hi
```

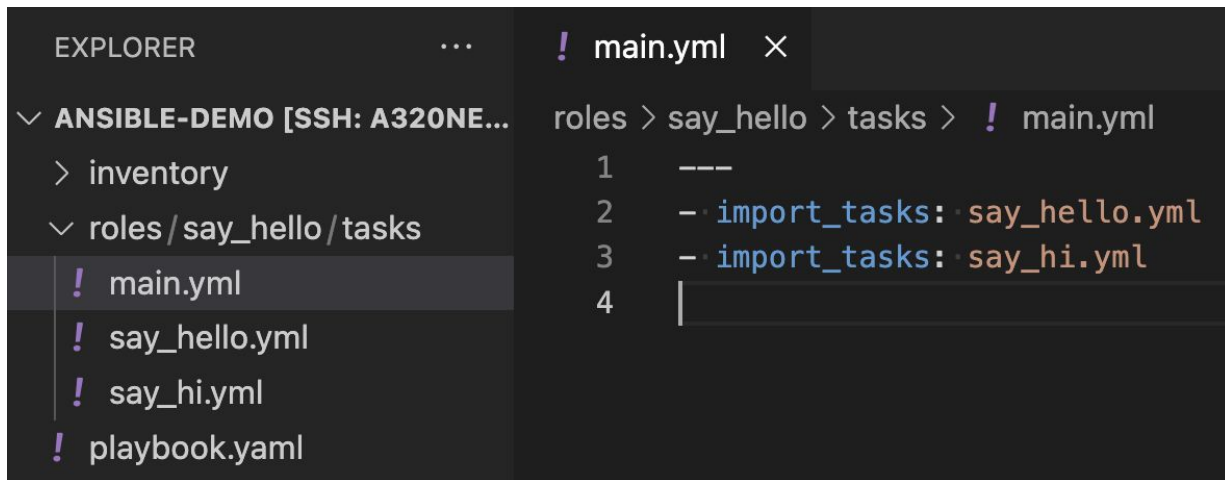
Conditional Execution

- **when** keyword
- https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_conditionals.html
- Run tasks only when a condition is satisfied

```
11  - name: Print OracleLinux version
12    debug:
13      msg: "We are on OracleLinux-{{ ansible_facts['distribution_major_version'] }}"
14    when: ansible_facts['distribution'] == "OracleLinux"
```

Import Tasks

- Split roles into multiple files to keep the codebase clean
- Roles are imported and executed in the defined order
- https://docs.ansible.com/ansible/latest/collections/ansible/builtin/import_tasks_module.html



The screenshot shows an IDE with two panels. The left panel, titled 'EXPLORER', displays a file tree for a project named 'ANSIBLE-DEMO [SSH: A320NE...]'. The tree structure is as follows:

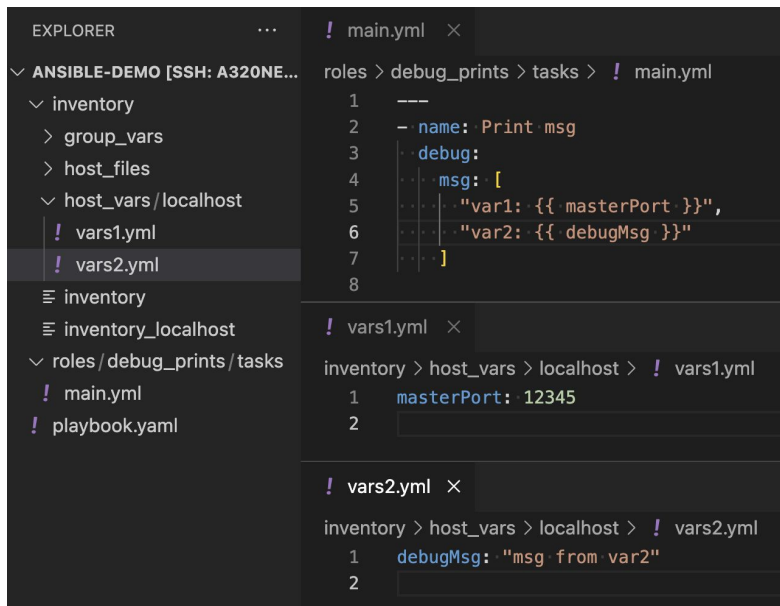
- ANSIBLE-DEMO [SSH: A320NE...]
 - inventory
 - roles / say_hello / tasks
 - ! main.yml (selected)
 - ! say_hello.yml
 - ! say_hi.yml
 - ! playbook.yml

The right panel shows the content of the selected file, 'main.yml', with the following YAML code:

```
roles > say_hello > tasks > ! main.yml
1 ---
2 - import_tasks: say_hello.yml
3 - import_tasks: say_hi.yml
4 |
```


Host Vars

- Define variables for each host in a specific file
<prevPath>/inventory/host_vars/<host>/*.yaml



The screenshot shows an IDE with three panels. The left panel is the Explorer, showing a project structure for 'ANSIBLE-DEMO [SSH: A320NE...]'. It includes folders for 'inventory' (with subfolders 'group_vars', 'host_files', and 'host_vars/localhost') and 'roles/debug_prints/tasks'. The 'host_vars/localhost' folder contains 'vars1.yml' and 'vars2.yml'. The right panel shows the content of these files. The top editor shows 'main.yml' with a task 'Print msg' that uses variables from 'vars1.yml' and 'vars2.yml'. The bottom editor shows 'vars1.yml' with 'masterPort: 12345' and 'vars2.yml' with 'debugMsg: "msg from var2"'. The breadcrumb for the bottom editor is 'inventory > host_vars > localhost > ! vars2.yml'.

```
EXPLORER
...
! main.yml x
roles > debug_prints > tasks > ! main.yml
1 ---
2 - name: Print msg
3   debug:
4     msg: [
5       "var1: {{ masterPort }}",
6       "var2: {{ debugMsg }}"
7     ]
8

! vars1.yml x
inventory > host_vars > localhost > ! vars1.yml
1 masterPort: 12345
2

! vars2.yml x
inventory > host_vars > localhost > ! vars2.yml
1 debugMsg: "msg from var2"
2
```

Host Vars

- https://docs.ansible.com/ansible/latest/inventory_guide/intro_inventory.html#host-variables

```
[plnyrich@a320neo ansible-demo]$ ansible-playbook -i inventory/inventory_localhost playbook.yaml

PLAY [all] *************************************************************************************************************************************

TASK [Gathering Facts] *************************************************************************************************************************************
[WARNING]: Platform linux on host localhost is using the discovered Python interpreter at /usr/bin/python3.9, but future
installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.15/reference_appendices/interpreter_discovery.html for more information.
ok: [localhost]

TASK [debug_print : Print msg] *************************************************************************************************************************************
ok: [localhost] => {
  "msg": [
    "var1: 12345",
    "var2: msg from var2"
  ]
}

PLAY RECAP *********************************************************************
localhost                : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[plnyrich@a320neo ansible-demo]$
```

Group Vars

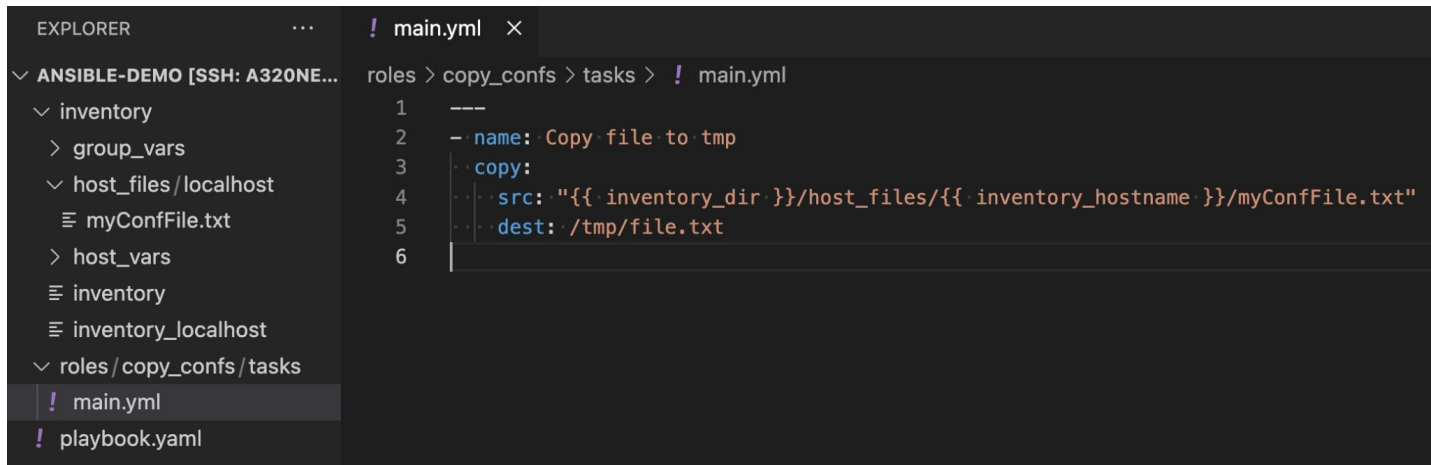
- Similar as host vars but we can define variables for groups
<prevPath>/inventory/group_vars/<group>/*.yml

```
inventory > ≡ inventory
1    [web_servers]
2    web1.company.com
3    web2.company.com
4
5    [db_servers]
6    db1.company.com
7    db2.company.com
8    
```

```
▼ ANSIBLE-DEMO [SSH: A320NE...]
▼ inventory
  ▼ group_vars
    ▼ db_servers
      ! vars.yml
    ▼ web_servers
      ! vars.yml
  > host_files
  > host_vars
  ≡ inventory
  ≡ inventory_localhost
```

Host Files

- Not directly in Ansible, but handy
`<prevPath>/inventory/host_files/<host>/*.yaml`
- Reference file from role:
`{{ inventory_dir }}/host_files/{{ inventory_hostname }}/file`



The screenshot shows an IDE with a file explorer on the left and a code editor on the right. The file explorer, titled 'EXPLORER', shows a project named 'ANSIBLE-DEMO [SSH: A320NE...]'. The tree structure includes 'inventory' (with sub-items 'group_vars' and 'host_files/localhost' containing 'myConfFile.txt'), 'host_vars', 'inventory', 'inventory_localhost', and 'roles/copy_confs/tasks'. The 'roles/copy_confs/tasks/main.yml' file is selected. The code editor shows the content of 'main.yml' with the following task:

```
1 ---
2 - name: Copy file to tmp
3   copy:
4     src: "{{ inventory_dir }}/host_files/{{ inventory_hostname }}/myConfFile.txt"
5     dest: /tmp/file.txt
6
```

Vaults

- Storing sensitive information
- Encryption of variables, roles, whole files, ...
- https://docs.ansible.com/ansible/latest/vault_guide/vault.html
- <https://www.digitalocean.com/community/tutorials/how-to-use-vault-to-protect-sensitive-ansible-data>

Running shell

- Use **shell** module
- Use pipe (|) for multiline string input
- https://docs.ansible.com/ansible/latest/collections/ansible/builtin/shell_module.html

```
roles > shell > tasks > ! main.yml
1  ---
2  - name: Run shell command
3    shell: |
4      date
5      sleep 1
6      exit 0
7    register: shell_cmd
8
9  - name: Print shell command output
10   debug:
11     msg: "stdout: {{ shell_cmd.stdout }}"
12
```

Part 5 - Demo

What do we want to do?

- Quickly deploy web server with our website
 - Install dependencies
 - Install web server and configure web server
 - Open port in firewall
 - Start the web server
 - Access our website

Result

