ACS GT Training 2025 Hands-on Labs - Day 2

LAB03 - Ansible

The aim for this lab is to demonstrate the basics of an Ansible workflow. We will use the VMs created in the previous labs and we will install a web server and MySQL server. We will configure Windows server, add users and sinstall IIS. In later labs we will use the similar playbooks with vLM.

Start by installing Ansible directly into your machine

Task 3 - Basic Ansible command and Inventory

We will run some basic Ansible commands to get the basics how Ansible works

1. Create an inventory file (add the IP address for each server on its own line)
   a. Make sure you are in the root folder of your **crp-training** repository
   b. Create the file named `inventory`
      `nano inventory`
   c. Add your VM1, VM10, VM2 and VM20 IP addresses to the inventory file
      `10.180.144.1XX`
      `10.180.144.1XX+30`

`10.180.144.2XX`
`10.180.144.2XX+30`

For example for Student01 the list would be like
`10.180.144.101`
`10.180.144.131`
`10.180.144.201`
`10.180.144.231`

d. Save the file

2. Lets test if Ansible is working. Lets run a command named ansible, with options:
   a. all
      Specifies the list of hosts in inventory file we will use with the command
   b. --key-file
      Specifies the SSH private key Ansible uses to connect to the target host
   c. -i inventory
      Specifies the inventory file Ansible uses for hosts connection information
   d. -m ping
      Specifies Ansible module to run. We will use three different modules later in this task:
      i. `ping`
      ii. `win_ping`
      iii. `gather_facts`
   e. Run the full command:
      `ansible all --key-file ~/.ssh/id_ed25519 -i inventory -m ping`
3. **The command above must fail**. The reason is that when running Ansible commands from the command line, it will use the logged on user as the user to connect to the remote hosts. Lets modify our inventory file to add variables which specify the correct username for SSH. Add `ansible_user=root` behind every Linux host and `ansible_user=administrator ansible_shell_type=powershell`
   For Windows host we also need to specify the shell type Ansible uses, and it must be specified as powershell.
   The result will look similar:

```
10.180.144.1XX ansible_user=root
10.180.144.1XX+30 ansible_user=root
10.180.144.2XX ansible_user=administrator ansible_shell_type=powershell
10.180.144.2XX+30 ansible_user=administrator ansible_shell_type=powershell
```

4. Try again the Ansible command:
   `ansible all --key-file ~/.ssh/id_ed25519 -i inventory -m ping`
5. You will see the command runs successfully only on Linux host and fails on Windows hosts. Reason is, the the Ansible module `ping` works only with Linux hosts
6. Try command
   `ansible all --key-file ~/.ssh/id_ed25519 -i inventory -m win_ping`
7. This command will succeed on Windows hosts but fails on Linux. The reason is the same, the module `win_ping` works only with Windows hosts
8. Finally lets try module that works on both Linux and Windows hosts - gather_facts
   `ansible all --key-file ~/.ssh/id_ed25519 -i inventory -m gather_facts`
   You will see detailed information regards all the hosts
9. Now lets improve our inventory file, so we can target different hosts better. For this we will add groups into the inventory file.
   We will add the following groups: `linux`, `windows`, `web_servers`, `db_servers`, `file_servers`.
   We will add VM1 and VM10 member of `linux` group
   VM2 and VM20 member of `windows` group

VM1 and VM2 member of `web_server` group
VM10 member of `db_servers` group
VM20 member of `file_servers` group

10. Also we can organize the variables into different groups. Take a look at the result below
11. The result of the inventory file will look similar (Student with ID 01)

```
[linux]
10.180.144.101
10.180.144.131

[linux:vars]
ansible_user=root

[windows]
10.180.144.201
10.180.144.231

[windows:vars]
ansible_user=administrator
ansible_shell_type=powershell

[web_servers]
10.180.144.101
10.180.144.201

[db_servers]
10.180.144.131

[file_server]
10.180.144.231
```

12. Notice that one host can be member of different groups
13. Now lets try to run the same basic Ansible commands against the host groups
    `ansible windows --key-file ~/.ssh/id_ed25519 -i inventory -m win_ping`
    `ansible linux --key-file ~/.ssh/id_ed25519 -i inventory -m ping`
14. Both commands will now run successfully as these are targeted against correct operating systems

Task 4 - Ansible configuration file

We will generate Ansible configuration file, in order to set some defaults. In this way we can improve and simplify our Ansible commands

1. Create Ansible config file. Make sure you are in the root folder of your repository
   `nano ansible.cfg` . This ansible configuration file will be used for default configuration settings however it is possible to have multiple hierarchical configuration files within an ansible environment. This will not be discussed discussed further during this tutorial.
2. Paste the following contents

```
[defaults]
inventory = inventory
private_key_file = ~/.ssh/id_ed25519
```

3. Now you can run the ansible commands much easier
   `ansible windows -m win_ping`
   `ansible linux -m ping`

Task 5 - Install requirements

In this task we will install some additional Ansible modules in order write our first playbooks. We can do that automatically with `ansible-galaxy` command

1. Generate new file into the root of your repository
   `nano requirements.yml`
2. Paste the following content into the file and save it
   https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/requirements.yml

```
---
collections:
  - name: ansible.posix
    version: "1.5.4"
    source: https://galaxy.ansible.com
  - name: ansible.windows
    version: "1.14.0"
    source: https://galaxy.ansible.com
  - name: community.general # for yaml output
    version: "7.0.1"
    source: https://galaxy.ansible.com
  - name: community.windows
    version: "1.13.0"
    source: https://galaxy.ansible.com
  - name: chocolatey.chocolatey
    version: "1.4.0"
    source: https://galaxy.ansible.com
```

3. Install Requirements.
   `ansible-galaxy install -r requirements.yml`

Task 6 - Ansible Playbooks

We have now reached to the state when we can really start using the power of Ansible. This is writing playbooks. Playbook in Ansible is series of plays (tasks) that we want to achieve on the targets

1. Lets write our first playbook, that updates the apt cache on all Linux hosts and install nginx on Linux web server.
2. Generate the playbook file into the root of the repository playbook.yml
   `nano playbook.yml`
3. Paste the content and save the file
   https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/playbook1.yml

```
---

- hosts: linux
  tasks:
  - name: update repository index
    apt:
      update_cache: yes

- hosts: web_servers:&linux
  tasks:
  - name: install nginx
    ansible.builtin.apt:
      name: nginx
      state: present
```

4. Prior to running the playbook take a moment to consolidate you learning and understand where all of the steps that you have take so far link in.
5. Run the playbook with ansible-playbook command
   `ansible-playbook playbook.yml`
6. You should see the following output

```
jarkko@jaub:~/crp-training$ ansible-playbook playbook.yml

PLAY [linux] ***********************************************************************************************************************************

TASK [Gathering Facts] *************************************************************************************************************************
ok: [10.181.176.151]
ok: [10.181.176.121]

TASK [update repository index] *****************************************************************************************************************
changed: [10.181.176.121]
changed: [10.181.176.151]

PLAY [web_servers:&linux] **********************************************************************************************************************

TASK [Gathering Facts] *************************************************************************************************************************
ok: [10.181.176.121]

TASK [install nginx] ***************************************************************************************************************************
changed: [10.181.176.121]

PLAY RECAP *************************************************************************************************************************************
10.181.176.121             : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
10.181.176.151             : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

7. You can also try to access your webserver
   http://10.180.144.1XX (replace the IP address with your own host IP)

Task 7 – Improve your playbook

To target different hosts within our inventory, we have multiple options. One way is to specify **hosts** key in the playbook, like we did in previous example. Another way is to have **when** conditional in a task.

Within Task 3 Step 8 we used a an option to **gather_facts** which returned a substantial amount of information about the target VM. These "facts" can be used to specify the target environment in a much more granular way that we have previously used "Windows" and "Linux" from the inventory.

To aid understanding, execute the next commands and identify and search through the text for the values **"ansible_distribution"**. Typically this is found near the top of the script.

`ansible windows -m gather_facts`
`ansible linux -m gather_facts`

```
"ansible_distribution": "Ubuntu",
```

```
"ansible_distribution": "Microsoft Windows Server 2022 Standard",
```

Lets add **when: ansible_distribution == "Ubuntu"** to our Ubuntu specific tasks. In this way we can make sure that the play does not run on incorrect OS.

Also lets add IIS install to our Windows web server and disable the Windows Updates Service. For Windows tasks we will add **when: ansible_distribution == "Microsoft Windows Server 2022 Standard"** key

1. It would be more efficient to update and enhance the content of playbook.yml however for the purpose of reinforcing learning **create another playbook in the root of the repository called "playbook2.yml"** and add the following code:
   https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/playbook2.yml
   Added the following Windows tasks:

```
- hosts: windows
  tasks:
  - name: Disable Windows Updates Service
    win_service:
      name: wuauserv
      state: stopped
    when: ansible_distribution == "Microsoft Windows Server 2022 Standard"

- hosts: web_servers:&windows
  tasks:
  - name: install IIS
    win_feature:
      name: "Web-Server"
      state: present
      restart: yes
      include_sub_features: yes
      include_management_tools: yes
    when: ansible_distribution == "Microsoft Windows Server 2022 Standard"
```

2. Run the playbook with ansible-playbook command
   `ansible-playbook playbook2.yml`
3. Verify you can access your both webservers via browser and you can see the default websites
   a. 10.180.144.1XX+30 and 10.180.144.2XX

Task 8 – Copy files and Templates

We will now improve our nginx and IIS install playbook, by adding our own custom index files. We will add our own image and own default HTML page. We want to use Templates for the HTML page, as we want each host to have unique default page

1. Generate directory named “files” and “templates” in the root of the repository.
   `mkdir files`
   `mkdir templates`
2. Generate new file named “**default_site.html.j2**” within the **templates** folder and paste the following content:
   https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/default_site.html.j2
3. Upload any image you want to use on your webpage or upload the following to the “**files**” folder

   https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/my_image.jpg
4. Copy your playbook2.yml to playbook3.yml and modify your Linux web server in your playbook as follows:
   https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/playbook3.yml

```
  - name: Create images directory
    ansible.builtin.file:
      path: "/var/www/html/images"
      state: directory
      mode: '0755'
    when: ansible_distribution == "Ubuntu"

  - name: Upload image to webserver
    ansible.builtin.copy:
      src: my_image.jpg
      dest: "/var/www/html/images/my_image.jpg"
      mode: '0644'
    when: ansible_distribution == "Ubuntu"

  - name: Deploy custom index page
    ansible.builtin.template:
      src: default_site.html.j2
```

```
      dest: "/var/www/html/index.html"
      mode: '0644'
    when: ansible_distribution == "Ubuntu"

...

  - name: Copy index test page from template
    ansible.windows.win_template:
      src: default_site.html.j2
      dest: "C:\\inetpub\\wwwroot\\index.html"
    when: ansible_distribution == "Microsoft Windows Server 2022 Standard"

  - name: Create directory for images
    ansible.windows.win_file:
      path: C:\inetpub\wwwroot\images
      state: directory
    when: ansible_distribution == "Microsoft Windows Server 2022 Standard"

  - name: Copy a single file
    ansible.windows.win_copy:
      src: my_image.jpg
      dest: C:\inetpub\wwwroot\images
    when: ansible_distribution == "Microsoft Windows Server 2022 Standard"
```

5. Run the playbook with ansible-playbook command
   `ansible-playbook playbook3.yml`
6. Verify you can access your both webservers via browser and you can see the modified websites
   a. 10.180.144.1XX and 10.180.144.2XX
   b. Page refresh might be required

Task 9 - Creating a Simple Role

Our playbook is getting messy, and we are going to tidy it up with the help of roles. First we will generate new role that will install MySQL server. Then we will convert our current work into roles, that will install nginx and IIS.

1. The easiest way to generate the default role skeleton, is to use the `ansbile-galaxy` command
   As roles are located in the roles directory, we need to generate the **roles** folder into the root of the repository.
   `mkdir roles`
2. Make sure to change directory to `roles/`.
   `cd roles`
3. Use `ansible-galaxy` to create the role skeleton:
   `ansible-galaxy init install_mysql`
4. This will generate the default files and folders the Ansible is expecting for a role by default
5.
```
user@jaub:~/crp-training/roles$ ansible-galaxy init install_mysql
- Role install_mysql was created successfully
user@jaub:~/crp-training/roles$ ls
install_mysql
user@jaub:~/crp-training/roles$ ls install_mysql/
defaults  files  handlers  meta  README.md  tasks  templates  tests  vars
```

6. **Tasks**
   Update `tasks/main.yml` with the following:
   https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/mysql-tasks.yml

```
---
- name: Install MySQL package
```

```
  ansible.builtin.apt:
    name: mysql-server
    state: present

- name: Start MySQL service
  ansible.builtin.service:
    name: mysql
    state: started
    enabled: yes

- name: Configure MySQL
  ansible.builtin.template:
    src: my.cnf.j2
    dest: /etc/mysql/my.cnf
  notify: restart mysql
```

7. **Defaults**

   Update `defaults/main.yml` with the following:

```
---
mysql_root_password: default_root_password
mysql_bind_address: 127.0.0.1
mysql_port: 3306
```

8. **Templates**

   Add a new file into templates folder

   Add `templates/my.cnf.j2` with the following:

```
[mysqld]
user        = mysql
pid-file    = /var/run/mysqld/mysqld.pid
socket      = /var/run/mysqld/mysqld.sock
port        = {{ mysql_port }}
bind-address = {{ mysql_bind_address }}
```

9. **Handlers**

   Update `handlers/main.yml` with the following:
   https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/mysql-handler.yml

```
---
- name: restart mysql
  ansible.builtin.service:
    name: mysql
    state: restarted
```

10. This basic role covers the installation and starting of a MySQL server. You can expand it by adding more tasks for configuration, setting up users, or managing databases.
11. Roles can become quite complex, but they help in managing complex tasks and reusing code across different projects.
12. Generate new playbook file `main.yml` in the root of the **crp-training** folder by adding the file content:
    https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/main1.yml

```
---

- hosts: linux
  tasks:
  - name: update repository index
    apt:
      update_cache: yes
```

```
    when: ansible_distribution == "Ubuntu"

- hosts: windows
  tasks:
  - name: Disable Windows Updates Service
    win_service:
      name: wuauserv
      state: stopped
    when: ansible_distribution == "Microsoft Windows Server 2022 Standard"

- name: Database Servers
  hosts: db_servers
  roles:
    - install_mysql
```

13. Run the playbook to install MySQL server to VM10
    Run the playbook with ansible-playbook command
    `ansible-playbook main.yml`

Task 10 - Convert nginx and IIS tasks into roles

In this task we will cleanup the `playbook.yml` and convert the previous tasks of installing nginx and IIS into roles.

1. First lets create the folder structures for our new roles. Make sure you are in the roles folder and run
   `ansible-galaxy init install_nginx`
   `ansible-galaxy init install_iis`
2. Copy the following content into the `install_nginx/tasks/main.yml` file
   https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/main2.yml

```
---
- name: install nginx
  ansible.builtin.apt:
    name: nginx
    state: present
  when: ansible_distribution == "Ubuntu"

- name: Create images directory
  ansible.builtin.file:
    path: "/var/www/html/images"
    state: directory
    mode: '0755'
  when: ansible_distribution == "Ubuntu"

- name: Upload image to webserver
  ansible.builtin.copy:
    src: my_image.jpg
    dest: "/var/www/html/images/my_image.jpg"
    mode: '0644'
  when: ansible_distribution == "Ubuntu"

- name: Deploy custom index page
  ansible.builtin.template:
    src: default_site.html.j2
    dest: "/var/www/html/index.html"
    mode: '0644'
  when: ansible_distribution == "Ubuntu"
```

3. Copy the default_site.html.j2 from **templates** folder into `install_nginx/templates` folder
   `cp ../templates/default_site.html.j2 install_nginx/templates/`
4. Copy the **my_image.jpg** file into the `install_nginx/files` folder
   `cp ../files/my_image.jpg install_nginx/files/`
5. Add the following to the end of `main.yml` in the root folder of your repo
   https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/main4.yml

```
- name: nginx servers
  hosts: web_servers:&linux
  roles:
    - install_nginx

- name: iis Servers
  hosts: web_servers:&windows
  roles:
    - install_iis
```

6. Copy the default_site.html.j2 from **templates** folder into `install_iis/templates` folder
   `cp ../templates/default_site.html.j2 install_iis/templates/`
7. Copy the **my_image.jpg** file into the `install_iis/files` folder
   `cp ../files/my_image.jpg install_iis/files/`
8. Copy the following content into the `install_iis/tasks/main.yml` file
   https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/main3.yml

```
---
- name: install IIS
  ansible.windows.win_feature:
    name: "Web-Server"
    state: present
    restart: yes
    include_sub_features: yes
    include_management_tools: yes
  when: ansible_distribution == "Microsoft Windows Server 2022 Standard"

- name: Copy index test page from template
  ansible.windows.win_template:
    src: default_site.html.j2
    dest: "C:\\inetpub\\wwwroot\\index.html"
  when: ansible_distribution == "Microsoft Windows Server 2022 Standard"

- name: Create directory for images
  ansible.windows.win_file:
    path: C:\inetpub\wwwroot\images
    state: directory
  when: ansible_distribution == "Microsoft Windows Server 2022 Standard"

- name: Copy a single file
  ansible.windows.win_copy:
    src: my_image.jpg
    dest: C:\inetpub\wwwroot\images
  when: ansible_distribution == "Microsoft Windows Server 2022 Standard"
```

9. The final *crp-training/main.yml* should look as follows
   https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/main.yml
10. Run the final playbook
    `ansible-playbook main.yml`

Task 11 - Commit and push your changes back into the git origin

Now when we have finished our work, we should save our work back into the Git server

1. Make sure you are in the root of the repository

```
user@jaub:~/crp-training$ ls
ansible.cfg  files  inventory  main.yml  playbook2.yml  playbook.yml  README.md  requirements.yml  roles
templates
```

2. Run git status command to see what has been changed

   `git status`

   Should see the following output

```
user@jaub:~/crp-training$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    ansible.cfg
    files/
    inventory
    main.yml
    playbook.yml
    requirements.yml
    roles/
    templates/

nothing added to commit but untracked files present (use "git add" to track)

```

3. Run git pull command, to see if anybody else has committed into your repository in the meanwhile

   `git pull`

   Should see the following output, I expect the no one has pushed any updates into your repository

```
user@jaub:~/crp-training$ git pull
Username for 'https://git.int.i82.ste13.com': ste13.trg.gt<id>
Password for 'https://ste13.trg.gt<id>@git.int.i82.ste13.com':
Already up to date.
```

4. Lets add all our changes into the git commit list

   `git add .`

   Should get the following output if also running `git status`

```
user@jaub:~/crp-training$ git add .
user@jaub:~/crp-training$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   ansible.cfg
    new file:   files/my_image.jpg
    new file:   inventory
    new file:   main.yml
    new file:   playbook.yml
    new file:   playbook2.yml
    new file:   requirements.yml
```

```
    new file:   roles/install_iis/README.md
    new file:   roles/install_iis/defaults/main.yml
    new file:   roles/install_iis/files/my_image.jpg
    new file:   roles/install_iis/handlers/main.yml
    new file:   roles/install_iis/meta/main.yml
    new file:   roles/install_iis/tasks/main.yml
    new file:   roles/install_iis/templates/default_site.html.j2
    new file:   roles/install_iis/tests/inventory
    new file:   roles/install_iis/tests/test.yml
    new file:   roles/install_iis/vars/main.yml
    new file:   roles/install_mysql/README.md
    new file:   roles/install_mysql/defaults/main.yml
    new file:   roles/install_mysql/handlers/main.yml
    new file:   roles/install_mysql/meta/main.yml
    new file:   roles/install_mysql/tasks/main.yml
    new file:   roles/install_mysql/templates/my.cnf.j2
    new file:   roles/install_mysql/tests/inventory
    new file:   roles/install_mysql/tests/test.yml
    new file:   roles/install_mysql/vars/main.yml
    new file:   roles/install_nginx/README.md
    new file:   roles/install_nginx/defaults/main.yml
    new file:   roles/install_nginx/files/my_image.jpg
    new file:   roles/install_nginx/handlers/main.yml
    new file:   roles/install_nginx/meta/main.yml
    new file:   roles/install_nginx/tasks/main.yml
    new file:   roles/install_nginx/templates/default_site.html.j2
    new file:   roles/install_nginx/tests/inventory
    new file:   roles/install_nginx/tests/test.yml
    new file:   roles/install_nginx/vars/main.yml
    new file:   templates/default_site.html.j2
```

5. Commit added changes

`git commit -m "Ansible lab changes"`

```
user@jaub:~/crp-training$ git commit -m "Ansible lab changes"
[main 3718cc6] Ansible lab changes
 37 files changed, 609 insertions(+)
 create mode 100644 ansible.cfg
 create mode 100644 files/my_image.jpg
 create mode 100644 inventory
 create mode 100644 main.yml
 create mode 100644 playbook.yml
 create mode 100644 playbook2.yml
 create mode 100644 requirements.yml
 create mode 100644 roles/install_iis/README.md
 create mode 100644 roles/install_iis/defaults/main.yml
 create mode 100644 roles/install_iis/files/my_image.jpg
 create mode 100644 roles/install_iis/handlers/main.yml
 create mode 100644 roles/install_iis/meta/main.yml
 create mode 100644 roles/install_iis/tasks/main.yml
 create mode 100644 roles/install_iis/templates/default_site.html.j2
 create mode 100644 roles/install_iis/tests/inventory
 create mode 100644 roles/install_iis/tests/test.yml
 create mode 100644 roles/install_iis/vars/main.yml
 create mode 100644 roles/install_mysql/README.md
 create mode 100644 roles/install_mysql/defaults/main.yml
 create mode 100644 roles/install_mysql/handlers/main.yml
 create mode 100644 roles/install_mysql/meta/main.yml
 create mode 100644 roles/install_mysql/tasks/main.yml
```

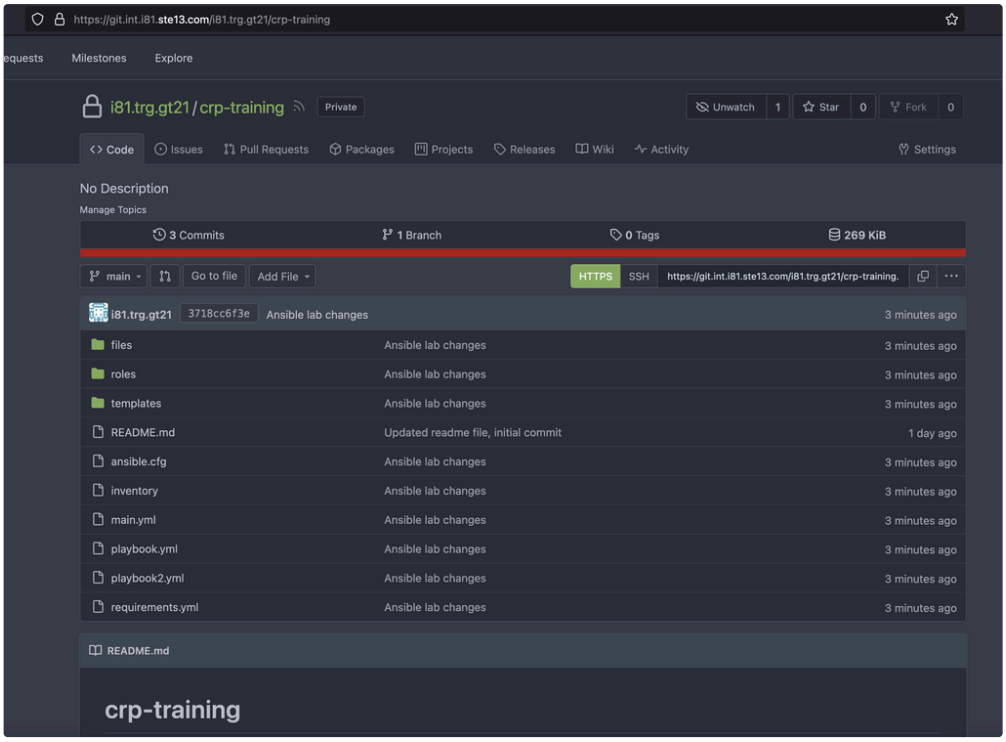
```
 create mode 100644 roles/install_mysql/templates/my.cnf.j2
 create mode 100644 roles/install_mysql/tests/inventory
 create mode 100644 roles/install_mysql/tests/test.yml
 create mode 100644 roles/install_mysql/vars/main.yml
 create mode 100644 roles/install_nginx/README.md
 create mode 100644 roles/install_nginx/defaults/main.yml
 create mode 100644 roles/install_nginx/files/my_image.jpg
 create mode 100644 roles/install_nginx/handlers/main.yml
 create mode 100644 roles/install_nginx/meta/main.yml
 create mode 100644 roles/install_nginx/tasks/main.yml
 create mode 100644 roles/install_nginx/templates/default_site.html.j2
 create mode 100644 roles/install_nginx/tests/inventory
 create mode 100644 roles/install_nginx/tests/test.yml
 create mode 100644 roles/install_nginx/vars/main.yml
 create mode 100644 templates/default_site.html.j2
```

6. Push the changes into the git origin

`git push origin main`

```
user@jaub:~/crp-training$ git push origin main
Username for 'https://git.int.i82.ste13.com': ste13.trg.gt<id>
Password for 'https://ste13.trg.gt<id>@git.int.i82.ste13.com':
Enumerating objects: 53, done.
Counting objects: 100% (53/53), done.
Delta compression using up to 2 threads
Compressing objects: 100% (28/28), done.
Writing objects: 100% (52/52), 246.21 KiB | 30.78 MiB/s, done.
Total 52 (delta 4), reused 0 (delta 0), pack-reused 0
remote: . Processing 1 references
remote: Processed 1 references in total
To https://git.int.i82.ste13.com/ste13.trg.gt<id>/crp-training.git
   31fa062..3718cc6  main -> main
```

7. You can also check your work from Gitea web UI



Task 12 - (EXTRA) Add role for the Windows fileserver

In this task there are no step-by-step guides. Try to add a new role, that installs the following to the Windows fileserver **10.180.144.2XX+30**

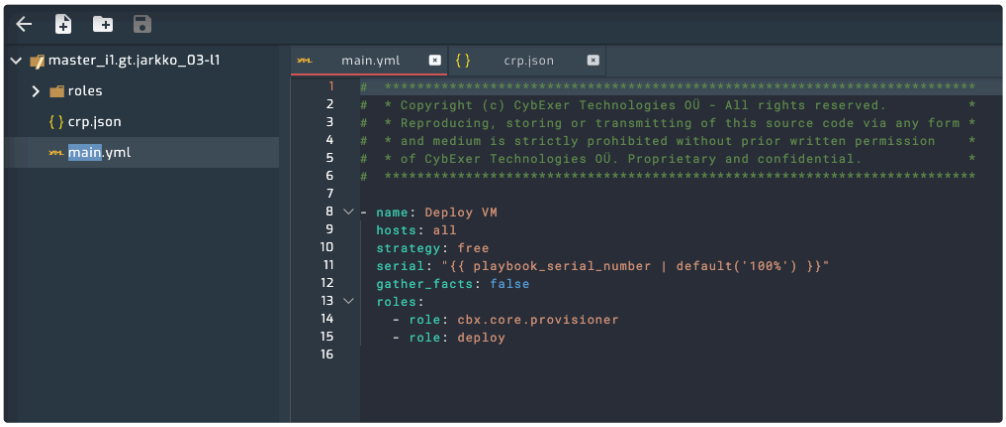
1. Add new user with name **share** and with password **Cool2Pass**
2. Generate a folder `C:\Public`
3. Share the folder as **Public**
4. Share must be accessible with the user **share**

LAB04 - vLM - Virtual Lab Manager

Task 1 - Create Linux System Definition

1. Login to **vCenter** https://vc.int.ste13.com to observe the activities in hypervisor.
2. Login to vLM using address https://vlm.int.ste13.com/
3. Select Platform **STE13-TRAINING**
4. Open "**System Definitions**" from the left panel.
5. Click "**New**" to start System definition creation wizard.
6. Use the following information to create your first Linux System definition (leave default values if not listed):
   a. GENERAL:
      i. System Definition name: ***<UserID>*-L1**
      ii. Description: **my first linux server**
      iii. Tags: "**LINUX**"
   b. CAPACITY:
      i. CPU: 1
      ii. Memory: 2048
      iii. Storage: 25GB
      iv. Nic count: 1
   c. ADVANCED:
      i. Golden Image: "**<UserID>-VM1-Template**" (Your generated template in Lab02)
      ii. Network Setup Method: "**netplan**"
      iii. Deploy API Option: "**VMWARE_SOAP**"
      iv. Ansible user: "**root**"
      v. Ansible password: "**Cool2Pass**"
      vi. Ansible Shell Type: "**sh**"
      vii. Ansible Become: "**true**"
      viii. Ansible Become user: "**root**"
      ix. Ansible Become Method: "**sudo**"
      x. Transport Protocol: "**ssh**"
   d. Guacomole Configurations → Add Config:
      i. Protocol: "**SSH**"
      ii. Port: "**22**"
      iii. Username {{ }}: "**root**"
      iv. Password {{ }}: "**Cool2Pass**"
7. Save the Definition
8. Go to **SYSTEM DEFINITIONS** rom the menu on the left and use the Search to find your definition. On your definition click **FILE EDITOR** and select **MASTER** version of the definition repository

9. You are presented with the repository of your System Definition files and folders



10. Lets leave it as a default for now, navigate back to the System Definitions page

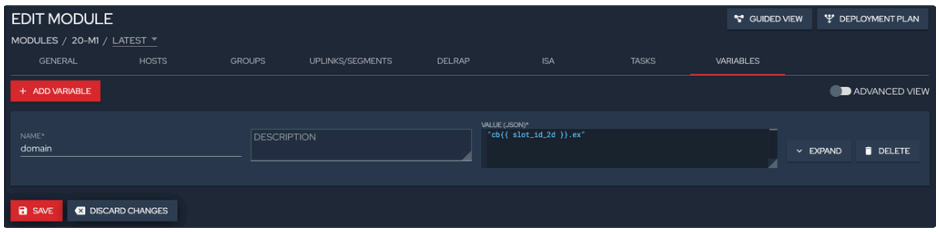
Task 2 - Create Windows System Definition

1. Click “**New**” to start System definition creation wizard.
2. Use the following information to create your first Windows System definition (leave default values if not listed):
   a. GENERAL
      i. System Definition name: ***<UserID>*-W1**
      ii. Description: **my first windows server**
      iii. Tags: “**WINDOWS**“
   b. CAPACITY:
      i. CPU: 1
      ii. Memory: 4096
      iii. Storage: 50GB
      iv. Nic count: 1
   c. ADVANCED
      i. Golden Image: “**<UserID>-VM2-Template**“ (Your generated template in Lab02)
      ii. Network Setup Method: “**powershell**“
      iii. Deploy API Option: “**VMWARE_SOAP**”
      iv. Ansible user: “**administrator**“
      v. Ansible password: “**Cool2Pass**“
      vi. Ansible Shell Type: “**powershell**“
   d. Guacomole Configurations → Add Config:
      i. Protocol: “**RDP**“
      ii. Port: “**3389**“
      iii. Username {{ }}: “**administrator**“
      iv. Password {{ }}: “**Cool2Pass**“
   e. Guacomole Configurations → Add Config:
      i. Protocol: “**SSH**“
      ii. Port: “**22**“
      iii. Username {{ }}: “**administrator**“
      iv. Password {{ }}: “**Cool2Pass**“
3. Save the Definition
4. Go to **SYSTEM DEFINITIONS** rom the menu on the left and use the Search to find your definition. On your definition click **FILE EDITOR** and select **MASTER** version of the definition repository
5. You are presented with the repository of your System Definition files and folders

6. Review the files and navigate back to the System Definitions page

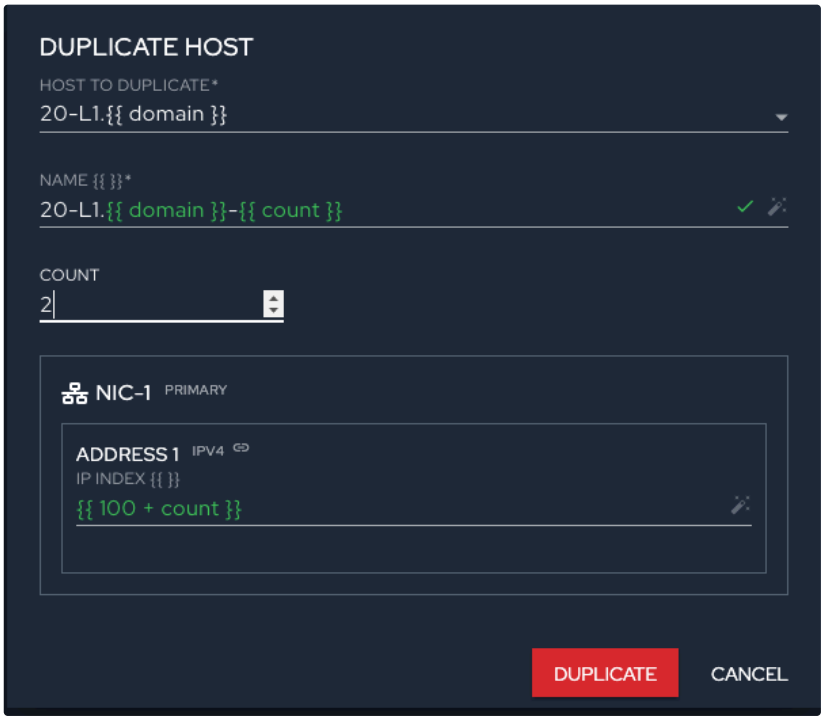
Task 3 - Generate vLM learning module

1. Open “**MODULES**“ on the left panel
2. Click “**NEW**“
3. Use the following information to create your Module (leave default values if not listed):
   a. General:
      i. Module Name: “**<User ID>-M1**“
      ii. Description: “**my learning module**“
      iii. Environment type: “**TRAINING**“
      iv. Click “**SAVE**”
   b. Variables:
      i. Click “**+ ADD VARIABLE**“
      ii. Add variable named “**domain**“ with value: `"cb{{ slot_id_2d }}.ex"`



   c. Uplink/Segments:
      i. Click “**+ Add Network**“
      ii. Name: “**NET-01**“
      iii. Type: “**SEGMENT**“
      iv. Network Type: “**DEDICATED**“
      v. Click “**SAVE**“
   d. Add one more network:
      i. Click “**+ Add Network**“
      ii. Name: “**NET-02**“
      iii. Type: “**SEGMENT**“
      iv. Network Type: “**DEDICATED**“
      v. Click “**SAVE**“
   e. Hosts:
      Lets add our system hosts we generated in previous steps. First lets add the Linux host:
      i. Click “**+ ADD SYSTEM HOST**“
      ii. Name: **<User ID>-L1.{{ domain }}**
      iii. System: “**<User ID>-L1**” (Your Linux System Definition)
      iv. Slot Usage = **Dedicated**
      v. NICS - > NIC1
         1. Network: “**NET-01**“
         2. Addresses
            a. Click “**+ ADD**“
            b. IPv4 IP Index: “**100**“
      vi. Click “**SAVE**“ and navigate back to the Modules page
   f. Lets add few more copies of the same Linux host:
      i. Find your Module and click “**DUPLICATE**”

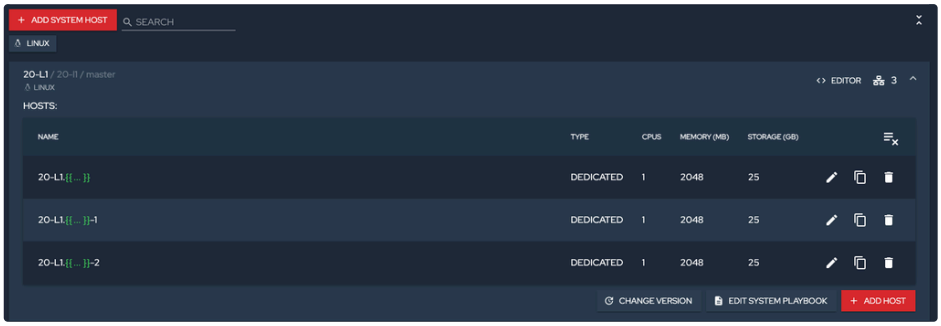
ii. Set **Count = 2** and click **“DUPLICATE“**



iii. Edit your new hosts in the module to verify that all information is already pre-filled, including IP addresses

g. Now lets add the Windows host to the module:

i. Click “**+ ADD SYSTEM HOST**“



ii. Name: “**<User ID>-W1.{{ domain }}**“

iii. System: “**<User ID>-W1**” (Your Windows System Definition)

iv. Slot Usage = **Dedicated**

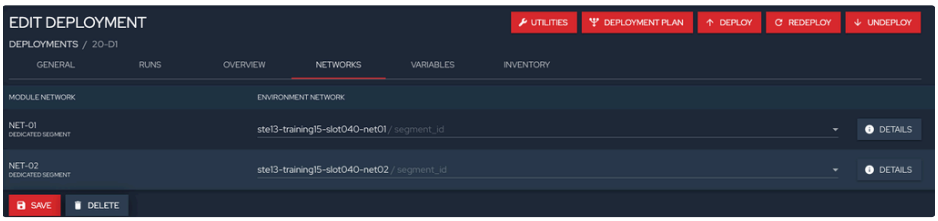
v. NICS - > NIC1

1. Network: “**NET-02**“
2. Addresses
    a. Click “**+ ADD**“
    b. IPv4 IP Index: “**100**“

vi. Click “**SAVE**“

Task 4 - Generate vLM Deploy item from the Module

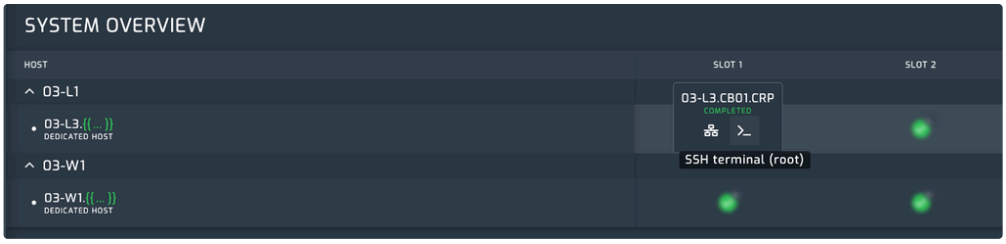
1. Open “**DEPLOYMENTS**“ on the left panel
2. Click “**+ NEW**“
3. Use the following information to create your Deploy (leave default values if not listed):
   a. General:
      i. Deployment Name: “**<User ID>-D1**“
      ii. Select Module: “**<User ID>-M1**“ (Your Module generated in previous lab)
      iii. Select Version: “**MASTER**“
      iv. Select Environment: “**STE13-TRAINING15**“
      v. Number of Slots: “**2**“
      vi. First Slot: **<User ID> x 2**
      (for user with ID 01 will start from Slot 02 and user with ID 12 will start from Slot 24)
      vii. Click “**SAVE**“
   b. Networks:
      i. Map Module network “**NET-01**“ to networks **ste13-training15-slot0(<User ID>x2)-net01.**



      ii. Map Module network “**NET-02**“ to networks **ste13-training15-slot0(<User ID>x2)-net02**.
      iii. To see more details about the connecting network, click the **DETAILS** button
   c. Click “**SAVE**“

Task 5 - Deploy the vLM Module

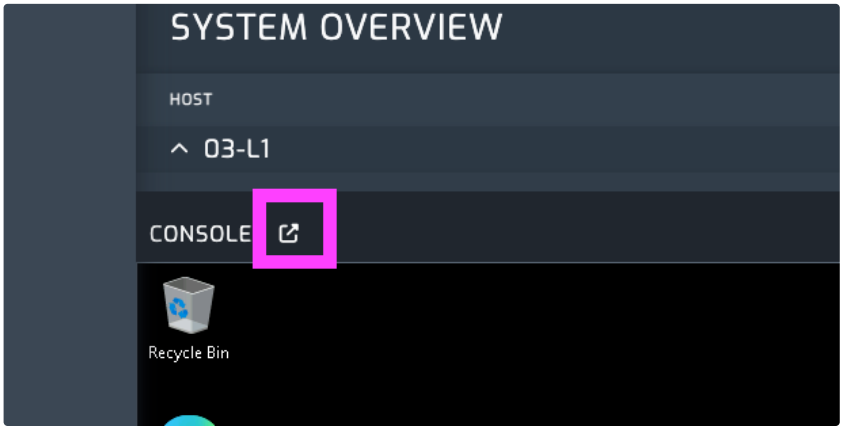
1. Open your Deployment “**<User ID>-D1**“
2. Click “**DEPLOY**“
3. Note that Deployment mode is: “**DEPLOY**“ and all systems in all slots are automatically selected.
4. Click “**DEPLOY**“ button
   We will deploy the empty systems to familiarise with the workflow
5. Observe the **Deployment Run Overview** page
   a. Expand your system deployment sections and see how Ansible is executing its plays and shows the result of each play in vLM
   b. To see more detailed log, click on the “**Container Logs**“ button, this outputs the whole ansible-runner Docker container logs into vLM
6. Switch to vSphere Client and observe the running “**Clone virtual machine**” tasks.
7. Cloning tasks might take some time to complete
8. Expand vSphere folders “**STE13-WORKLOADS → STE13-TRAINING15 → STE13-TRAINING15-SLOT0<User ID>x2**” from **vCenter** inventory tree and note your VMs:
   ***<User ID>*-L1.cbXX.ex**
   ***<User ID>*-L1.cbXX.ex-1**
   ***<User ID>*-L1.cbXX.ex-2**
   ***<User ID>*-W1.cbXX.ex**
9. Go back to vLM, go to “**DEPLOYMENTS**“ page and open your deployment.
10. Go to the **OVERVIEW** section and notice the targets status in the Slots. You can open console access directly in the view



a. Open the console for any of the Linux machines and any of the Windows machines.

b. You can use VMware vSphere console and native OS connection, either SSH or RDP. Try out both. You can also use SSH for Windows

11. Verify your Linux config:
    a. Type `ip a` for ip-address verification
    b. Type `hostname` for hostname verification
12. For Windows click the pop-out button to make the window detached from main vLM window.



13. Verify that copy/paste works, if you choose the RDP protocol. Copy/paste does not work over vSphere console connection

Task 6 - Add services to your Linux target VM

You will add a nginx web service to your VM with vLM. We will use the same Ansible playbooks that we used during **LAB03.** We will replicate the similar setup that we did with Ansible Core in LAB03.

1. Switch to vLM browser window
2. Select "**SYSTEM DEFINITIONS**" from left panel
3. Filter the page with your ID to show only your definitions
4. Use the **FILE EDITOR** to edit your created Linux VM definition "***<UserID>*-L1**"
5. Choose "**feature**" branch as the definition version (Previous file editor session from branch master)
6. Lets add a new role called "**install_nginx**"
    a. Under the roles folder generate new folder named: "**install_nginx**"
    b. Under the new install_nginx folder generate new folders named: "**tasks**", "**files**" and "**templates**"
    c. Under the tasks folder generate new file named: "**main.yml**"
    d. Enter the content of the file:
    https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/main2.yml

```
---
- name: Update and upgrade apt packages
  ansible.builtin.apt:
    update_cache: yes
    upgrade: yes
- name: install nginx
  ansible.builtin.apt:
```

```
    name: nginx
    state: present
  when: ansible_distribution == "Ubuntu"

- name: Create images directory
  ansible.builtin.file:
    path: "/var/www/html/images"
    state: directory
    mode: '0755'
  when: ansible_distribution == "Ubuntu"

- name: Upload image to webserver
  ansible.builtin.copy:
    src: my_image.jpg
    dest: "/var/www/html/images/my_image.jpg"
    mode: '0644'
  when: ansible_distribution == "Ubuntu"

- name: Deploy custom index page
  ansible.builtin.template:
    src: default_site.html.j2
    dest: "/var/www/html/index.html"
    mode: '0644'
  when: ansible_distribution == "Ubuntu"
```

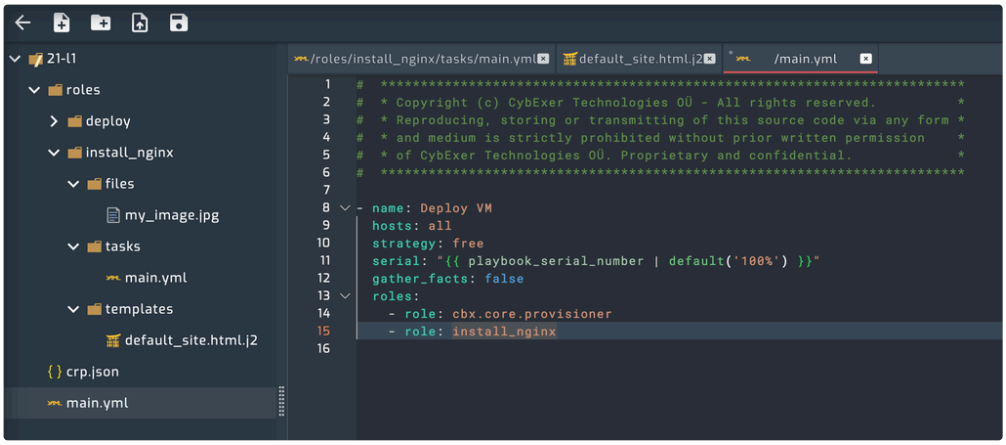
e. Save the file

f. Generate new file named "**default_site.html.j2**" with the **templates** folder and paste the following content: https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/default_site.html.j2

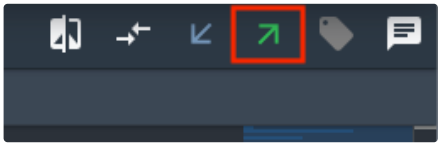
g. Save the file

h. Under the files folder upload new file named: "**my_image.jpg**" https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/my_image.jpg

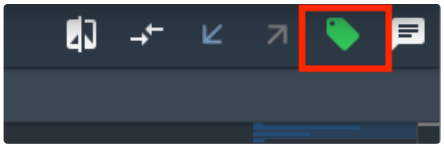
7. Add the new role to the main playbook: "**main.yml**" in the root folder
   a. Add value: "`- role: install_nginx`" after the Core Provisioner or Deploy role and **save** the file
8. The final result will look similar as follows:



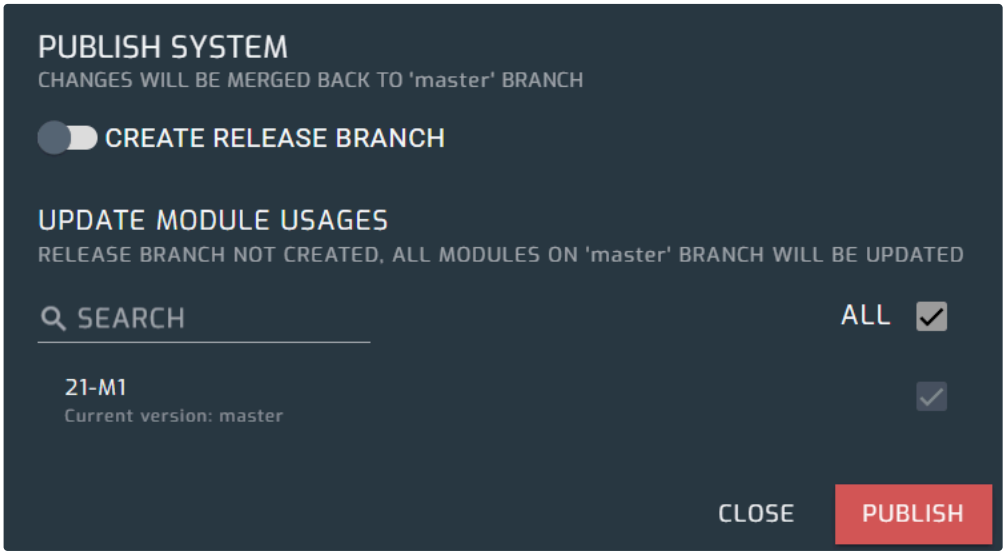
9. On the upper taskbar click first "**Push**" button



and then "**Publish**" button



10. On the “**Publish System Definition**“ modal **do not** create a release branch and select “**Update Module Usages**“ checkbox **ALL** and click “**Publish**“



11. Click “**Close**“ and navigate back to the “**System Definitions“** list

Task 7 - Add services to your Windows target VM

You will add the IIS role into the Windows target using vLM

1. Select “**SYSTEM DEFINITIONS**” from left panel
2. Filter the page with your ID to show only your definitions
3. Use **FILE EDITOR** to edit your created Windows VM definition “***<UserID>*-W1**”
4. Choose “**feature**“ branch as the definition version (Previous file editor session from branch master)
5. Lets add a new role called “**install_iis**“
   a. Under the roles folder generate new folder named: “**install_iis**“
   b. Under the new install_iis folder generate a new folder named: “**tasks**”, “**files**“ and “**templates**“
   c. Under the tasks folder generate new file named: “**main.yml**“
   d. Enter the content of the file:
   https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/main3.yml

```
---
- name: install IIS
  ansible.windows.win_feature:
    name: "Web-Server"
    state: present
    restart: yes
    include_sub_features: yes
    include_management_tools: yes
  when: ansible_distribution == "Microsoft Windows Server 2022 Standard"

- name: Copy index test page from template
  ansible.windows.win_template:
    src: default_site.html.j2
    dest: "C:\\inetpub\\wwwroot\\index.html"
  when: ansible_distribution == "Microsoft Windows Server 2022 Standard"

```

```
- name: Create directory for images
  ansible.windows.win_file:
    path: C:\inetpub\wwwroot\images
    state: directory
  when: ansible_distribution == "Microsoft Windows Server 2022 Standard"

- name: Copy a single file
  ansible.windows.win_copy:
    src: my_image.jpg
    dest: C:\inetpub\wwwroot\images
  when: ansible_distribution == "Microsoft Windows Server 2022 Standard"
```

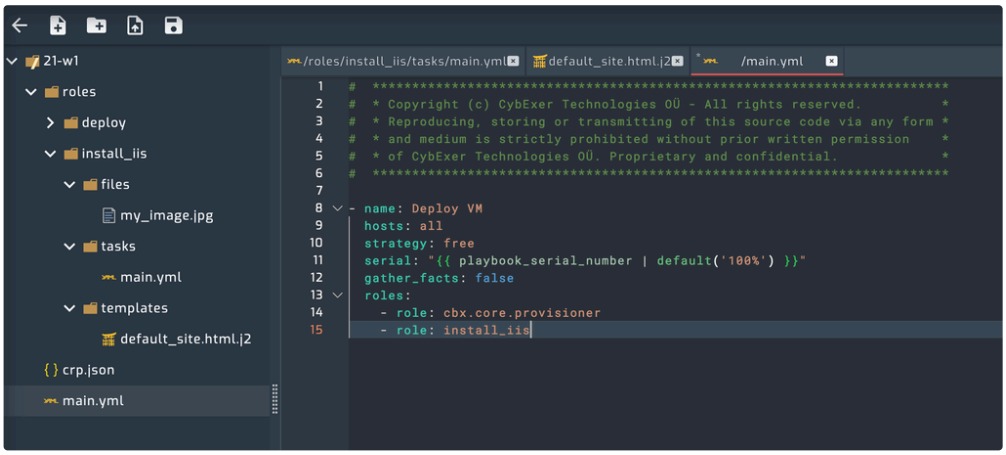
e. Save the file

f. Generate new file named "**default_site.html.j2**" to the **templates** folder and paste the following content: https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/default_site.html.j2

g. Save the file

h. Under the files folder upload new file named: "**my_image.jpg**" https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/2/my_image.jpg

6. Add the new role to the main playbook: "**main.yml**" in the root folder
7. Add value: "`- role: install_iis`" after the Core Provisioner role and **Save** the file
8. The final result will look as follows:

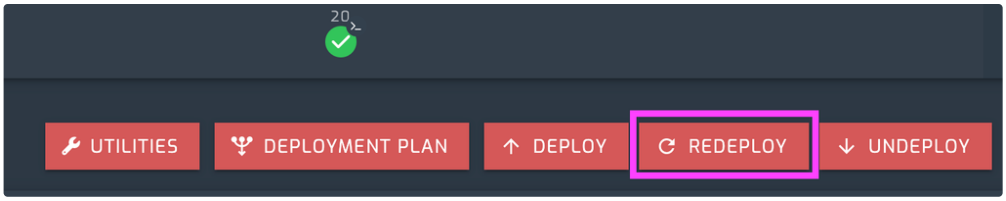


9. On the upper menubar click first "**Push**" button and then "**Publish**" button
10. On the "**Publish System Definition**" modal do no create a release branch and select "**Update Module Usages**" checkbox **ALL** and click "**Publish**"
11. Click "**Close**" and navigate back to the "**System Definitions**" list

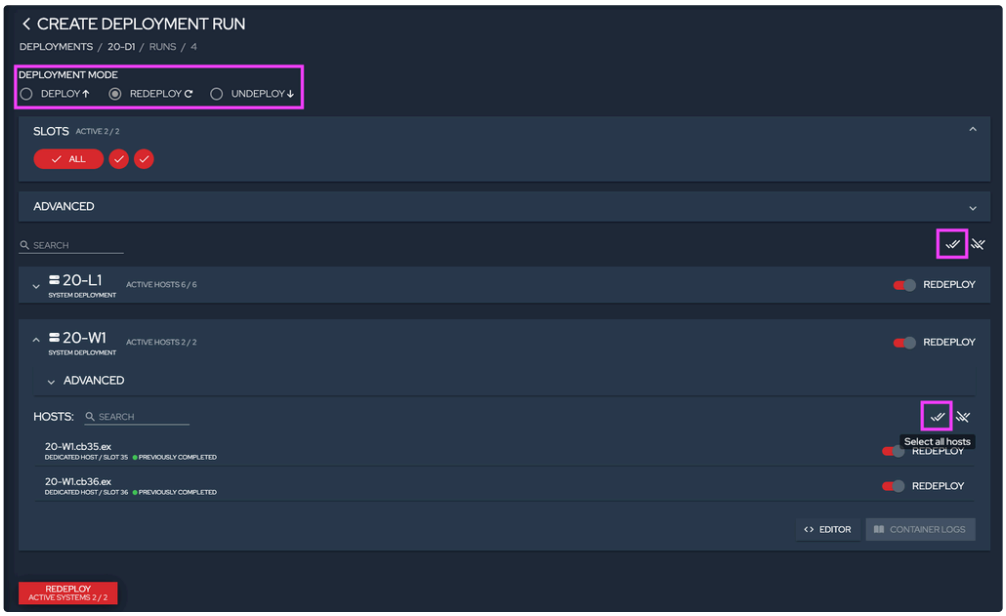
Task 8 - Redeploy your modified targets

In this lab we will use **REDEPLOY** feature. **REDEPLOY** will delete existing targets from hypervisor and will clone new targets from Golden images.

1. Open your Deployment "**<User ID>-D1**"
2. Click "**REDEPLOY**"



3. On the following page make sure your Deployment Mode is "**REDEPLOY**"
4. Enable the deploy on all hosts

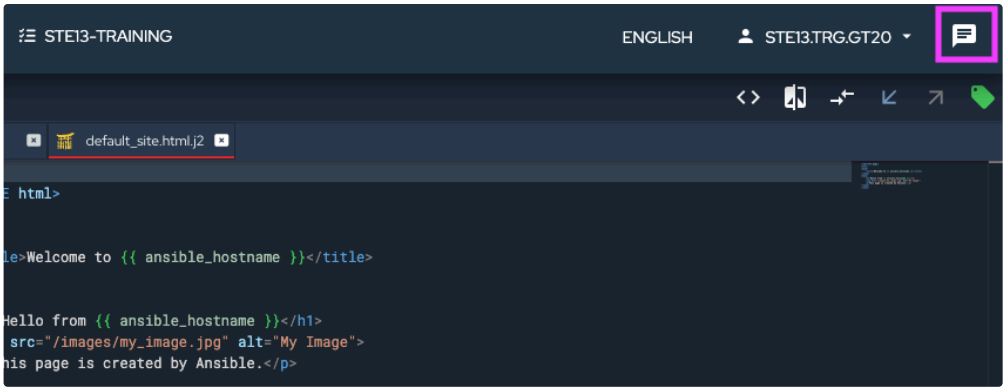


5. Click "**START**"
6. Verify that on **Deployment Run Overview** page your new steps are included.
7. When deploy finishes, check that you can access your new websites (**http://10.X.144.100**, **http://10.X.144.101** **and** **http://10.X.144.102**, **http://10.X.145.100**)

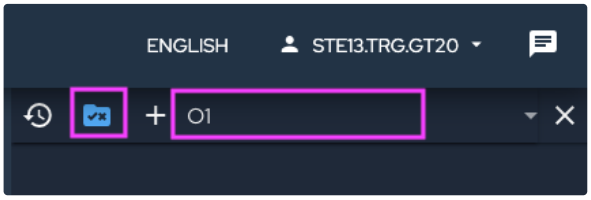
Task 9 - Add more features to your Web servers using AI

Lets add more data to our targets using built-in AI support

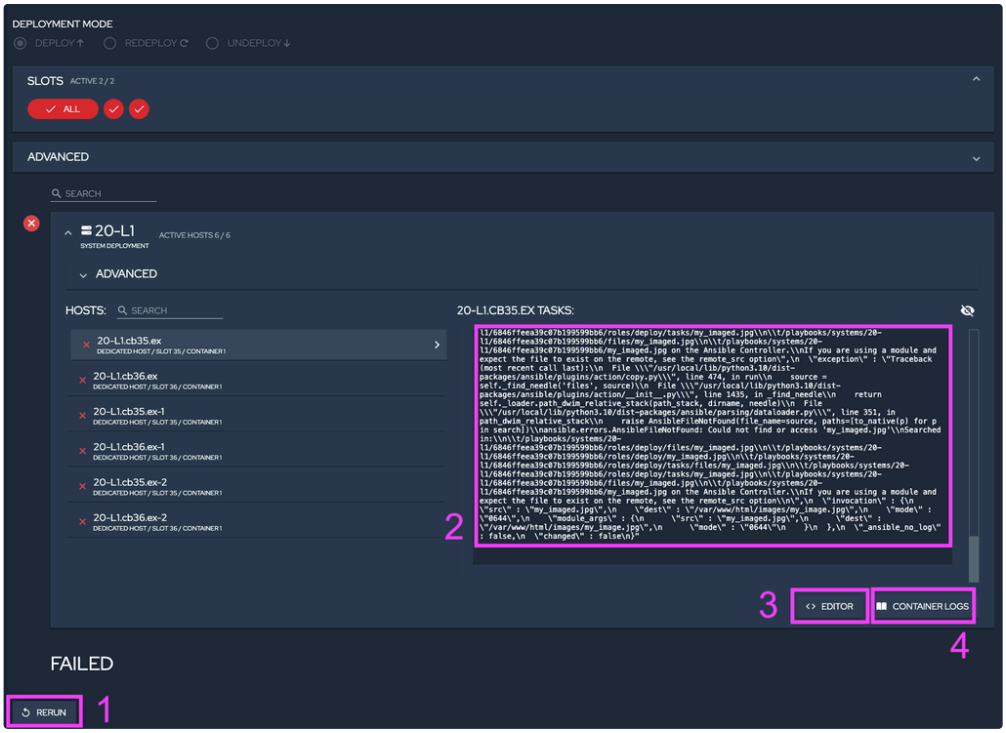
1. Select "**SYSTEM DEFINITIONS**" from left panel
2. Filter the page with your ID to show only your definitions
3. Use **FILE EDITOR** to edit your created Linux VM definition "***<UserID>*-L1**"
4. Choose "**feature**" branch as the definition version (Previous file editor session from branch master)
5. Navigate to roles → install_nginx → templates and edit file "**default_site.html.j2**"
6. Click on chat icon on top right



7. Allow chatbot to view and edit the repository. You can also choose your favourite language model.



   a. Ask the chatbot to modify your file. Try to ask questions like:
      ”Make the page look professional”
      ”Make the page look more beautiful”
8. Let the chatbot offer you the changes, and click “**Apply & Save**“ on the offered changes
9. When finished, “**Push**“ and “**Publish**“ your changes and make sure you also update the Module usages.
10. Do the similar actions on the Windows Web server
    a. Use **FILE EDIT** to edit your created Linux VM definition “***<UserID>*-W1**”
    b. Choose “**feature**“ branch as the definition version (Previous file editor session from branch master)
    c. Navigate to roles → install_iis → templates and edit file “**default_site.html.j2**“
    d. Allow chatbot to view and edit the opened file
    e. Again ask questions like:
       ”Make the page look more beautiful”
    f. Ask the chatbot to add more data, like:
       ”Add random online pictures to the page”
       ”Add something really cool on the page, go totally crazy”
11. Navigate to “**Deployments**“ and deploy all your targets
12. Now we will use the **DEPLOY** feature. During the DEPLOY, targets are not deleted, but changes are applied to the existing targets. This makes the change deployment much faster.
13. Make sure your Deployment Mode is “**DEPLOY**“. This mode skips the VM cloning and speeds up the reconfiguration in case the VM is already cloned.
14. Deploy **all** targets. You need to toggle **Enable Deploy** buttons
15. Monitor the deployment progress and check your targets after these are ready
16. Its common to encounter some syntax errors during content development. You can also ask AI to resolve errors in Ansible playbooks. When deploying multiple times, it's easier to use the RERUN feature of the failed deployment.



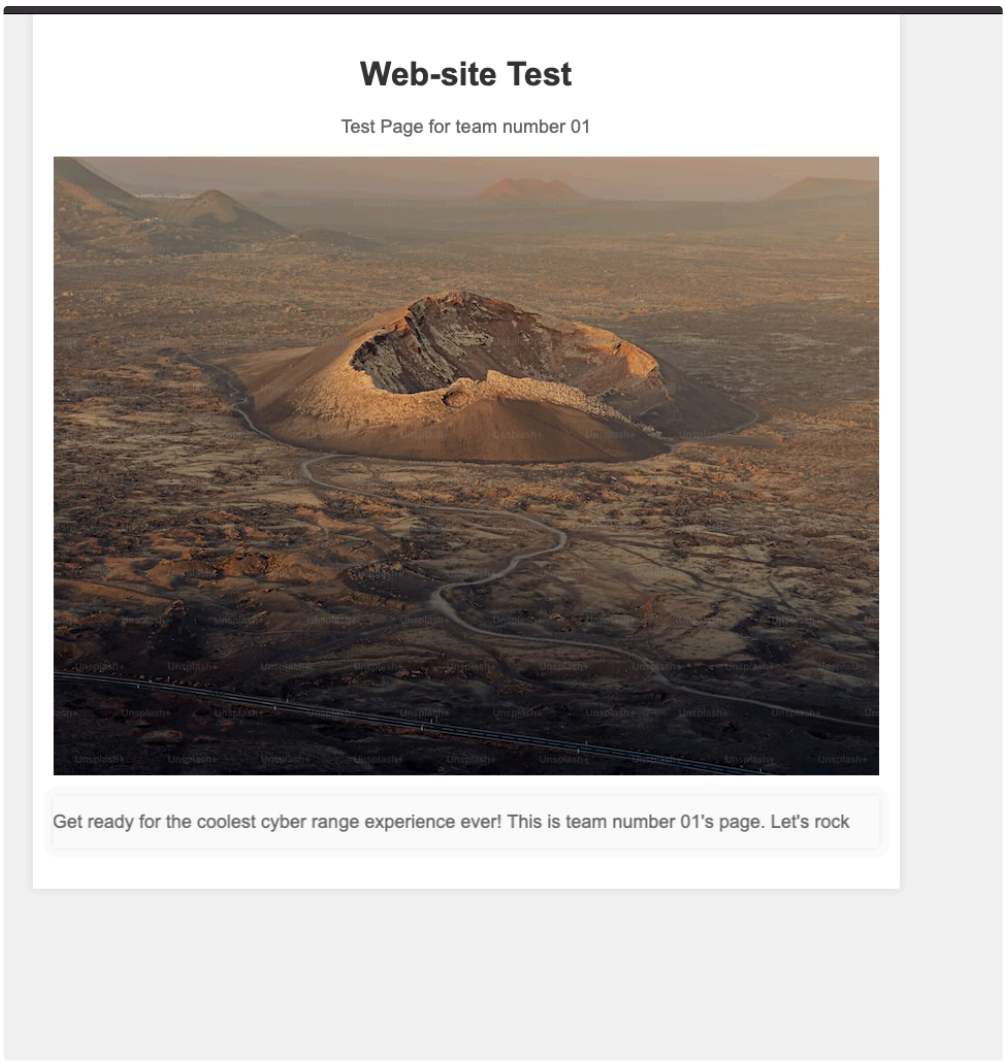
1 - Rerun will take you to the new deployment run view with the same options that were used during the failed run.

2 - Log can be copied to AI chat in the System Definition

3 - You can open the relevant System Definition quickly

4 - Sometimes the ansilbe-runner will not even start, if there are syntax errors in the playbooks. In those cases use the **CONTAINER LOGS** windows to observe where the error is

17. Eventually you should be able to get the modified website up and running.
18. Some examples of the results:



LAB05 – ISA

Task 1 – Create your own ISA exercise

1. Navigate to ISA with web browser https://isa.int.ste13.com
2. Log in with GT credentials
3. On the MODULES page you will see already existing exercises, please do not touch those at the moment
4. We will create new exercise, for this we need to have exercise template (in the form of .JSON file)
   a. Download example template: https://git.int.ste13.com/ste13.gt04/labs/src/branch/main/5/crp-training-exe.json
   b. **Edit lines 2** and **3** to change the exercise name according to your username
   c. you can browse around in rest of the file, but you dont need to change anything else there at the moment.
5. On the MODULES page in ISA click on **NEW MODULE** button
6. Paste your edited exercise template into the textbox and click **LOAD HYBIRD SCHEMA** and after this click **SUBMIT**
   a. It will take approximately a minute for all the data to be extracted into the system.
7. Start your exercise
   a. Click **START** on your exercise
   b. Is the exercise status "**Running**"?
8. Select your exercise and view its contents
   a. Click **SELECT** on your exercise

b. Browse the menu on the left
    i. MISSION BOARD
    ii. TASKS OVERVIEW

Task 2 – Use your ISA exercise

1. Log in to ISA as a **Red Team** member (`rt.user01` / `Untold_readded.9.4`)
    a. Select your exercise on the MODULES page
    b. Navigate to **Attack Campaign**
    c. Find one task and assign the task to attack BT-<User-ID> to yoursef
    d. Let's assume that you are finished with your attack, now click **Report** on the Attack Campaign page for BT-<User-ID>
    e. Notice that the report has been prefilled from template
    f. Generate Successful report
    g. Log out as a Red Team member
2. Log in to ISA as a **Blue Team** member (`bt<User-ID>.user01` / `Bursts_blight.8.2`)
    a. Select your exercise on the MODULES page
    b. Navigate to Mission Board and start Website Protecting task
    c. Write some answer and click Submit and Close
    d. Navigate to **Reporting** → **Situation Report** page and submit one Sitrep
    e. Log out as Blue Team member
3. Log in to ISA as a **White Team** member (`wt.user01` / `Decrypts_slidable.3.7`)
    a. Select your exercise on the MODULES page
    b. Navigate to **Judging** → **Attack Reports** view
    c. Analyse and Confirm the Red Team report
        i. Try to use auto assessment by AI
    d. Navigate to **Judging** → **Task Reports**
    e. Analyse and approve Blue Team task report
        i. Try to use auto assessment by AI
    f. Navigate to **Judging** → **Situation Reports**
    g. Analayse and grade the situation report
        i. Try to use auto assessment by AI
4. Observe different views
    a. Look at Smart City and see what cities are included
    b. Look at Campaign Live to see real-time status of the attacks and reports
    c. Look at Team Status to see if targets are up/down
    d. Look at Nice Leaderboard to see teams statistics regarding the Nice Framework
5. Try the After Action Report composer (AAR Composer)
    a. Log back in with your **GT** credentials
    b. Navigate to the **MODULES** page
    c. Find your exercise and **STOP** it
    d. Click on the ... menu
    e. Select AAR Composer
    f. Set some Title and **CREATE AAR**
    g. You can also download and change the template as you wish

Task 3 - Cleanup

Stop and Reset ISA

1. Navigate to Modules page and find your Exercise
2. Press the **STOP** button and confirm again with **STOP**
3. Click three dots menu ... and select RESET
4. Write **ERASE ALL** and **RESET**

Undeploy targets and delete Deployment in vLM

1. Navigate to **DEPLOYMENTS** and find your deployment: **<User ID>-D1**
2. Click **UNDEPLOY**, select all your targets and click **UNDEPLOY**
3. Wait while the task is finished. You can observe how all VMs are deleted from vSphere
4. Navigate to the **GENERAL** view of your deloyment and **DELETE** your deployment
5. All your data still exists in vLM so you can always regenerate the Deployment and deploy your Module