



DevOps

Training Assignment

Document Code	25e-BM/HR/HDCV/FSOFT
Version	1.1
Effective Date	00/00/2022

Hanoi, 01/2022

RECORD OF CHANGES

No	Effective Date	Change Description	Reason	Reviewer	Approver
1					

Contents

Day 1. Unit 1: Git	4
Assignment :	4
Day 2, 3. Unit2:Ansible	4
Assignment :	4
1. Install Ansible	4
2. Ansible Inventory	5
3. Ansible playbook	7
4. Role	12
5. Ansible galaxy	14
6. Ansible Tower	15
Day 4, 5. Unit 3: Terraform	14
Assignment :	14
1. Install Terraform and AWS CLI	14
2. Run the first time	16
3. AWS resource	19
4. Create infrastructure on AWS	22
Day 6, 7. Unit 4: CodePipeline	27
Assignment 1:	27
1. CodeCommit	27
2. Elastic Container Registry	32
3. CodeBuild	33
4. CodeDeploy	37
5. CodePipeline	42
Day 8, 9. Unit 5: Jenkins	46
Assignment :	46
1. Install Jenkins	46
2. Run the first Job in Jenkins	50
3. Manage Plugins	52
4. Manage Credentials	53
5. Jenkins pipeline	54
6. Run Jenkins pipeline with Ansible, Docker	58
7. Jenkins Master Slave	70
Day 8, 9. Unit 6: Prometheus+Grafana +AlertManager	74
Assignment :	74
1. Prometheus	74
2. Grafana	74
3. Alert Manager	74



CODE	:	Ansible
TYPE	:	Lab
LOC	:	<Lines of Code>
DURATION	:	2 day

Day 1. Unit 1: Git

Assignment :

- Install Gitbash, practice git cli

Objectives:

- Use Git competently

Problem Descriptions:

Assumptions:

Technical Requirements:

Questions to answer:

Estimated Time to complete: xx mins

Day 2, 3. Unit2:Ansible

Assignment :

1. Install Ansible

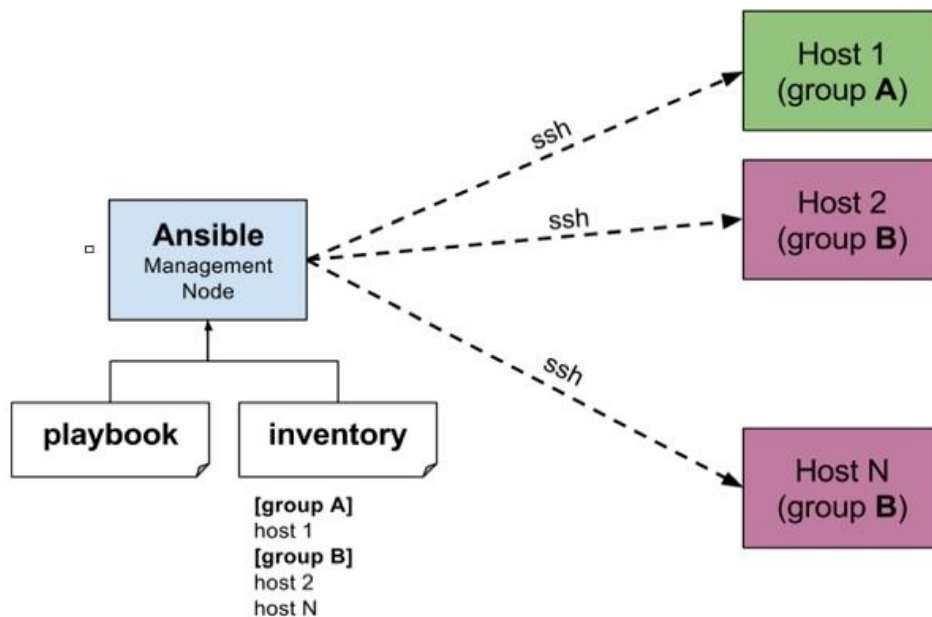
Cài đặt Ansible trên Ubuntu:

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install ansible
```

Có nhiều cách để gọi đến các server ta có thể copy public key của máy chạy ansible để máy server, cũng có thể sử dụng user, password để xác thực. Ở đây ta dùng keypair khi tạo server để xác thực, ta copy keypair vào /etc/ansible/ để sử dụng.

```
ubuntu@ip-172-31-29-124:/etc/ansible$ ll
total 64
drwxr-xr-x  3 root root 4096 Dec 11 13:06 ./
drwxr-xr-x 116 root root 4096 Nov 23 15:19 ../
-rw-r--r--  1 root root 1675 Nov 11 12:20 Training.pem
```

Mô hình hoạt động của Ansible:



2. Ansible Inventory

Đầu tiên ta cần khai báo host_group để dễ gọi các nhóm server. Ví dụ ta muốn cài đặt nginx lên nhóm server web_app, hay cài mysql lên nhóm server database, ... Các quy hoạch này của ansible giúp để gọi các lệnh về sau.

```
//nano /etc/ansible/hosts
[localhost]
127.0.0.1
[web_app]
172.31.13.222
[database]
172.31.13.200
```

Cấu trúc lệnh ansible như sau:

```
# ansible [hosts] -m [module] -a [tham số truyền vào] --private-key=[keypair]
```

Cấu trúc lệnh trên ta sử dụng private key để xác thực kết nối hoặc ta có thể sử dụng **username** và **password** để xác thực với cấu trúc lệnh sau:

```
#ansible [host] -m [module] -u [username]
```

Ví dụ, ta kiểm tra kết nối tới server ở nhóm **web_app** với lệnh sau:

```
ubuntu@ip-172-31-29-124:/etc/ansible$ ansible web_app -m ping --private-key=Training.pem
172.31.13.222 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
ubuntu@ip-172-31-29-124:/etc/ansible$
```

Ở đây ta sử dụng module **ping** để kiểm tra kết nối đến server nhóm **web_app** và sử dụng phương thức xác thực là dùng privatekey. Kết quả, kết nối thành công.

Trên đây là cách dùng truyền thống khi khai báo các tham số khi gọi lệnh. Ta có thể khai báo các tham số của server trong file hosts như sau:

```
[localhost]
127.0.0.1
[web_app]
172.31.13.222 ansible_ssh_user=ubuntu ansible_ssh_private_key_file=Training.pem
```

Ta thử chạy lại lệnh trên mà k cần khai báo lại private-key:

```
ubuntu@ip-172-31-29-124:/etc/ansible$ ansible web_app -m ping
172.31.13.222 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
```

ansible [tên host cần gọi] -m [tên module] -a [tham số truyền vào module]

-i : inventory host. Load thư viện host

-m : gọi module của ansible

-a : command_argument gửi kèm theo module mà ta đang gọi

-u : user

-vvvv : debug option

\$\$ ansible all -m ping (giải thích: gọi ping toàn bộ các hosts trong /etc/ansible/hosts)

\$\$ ansible all -m command -a uptime

\$\$ ansible all -a uptime (Default, ansible sẽ cho module = "command". Nên ta ko cần -m command thêm vào cũng được.)

\$\$ ansible -m shell -a 'top -bcn1 | head' (giải thích: chạy lệnh shell ở remote client!)(<https://images.viblo.asia/8335ed0c-7cfe-41b6-b7cf-bd37e05979a7.png>)

- restart mysql

\$\$ ansible dbserver -m service -a "name=mysql state=restarted" --key-file=~/.ssh/db.pem -u ubuntu --sudo

Tất cả module của ansible bạn có thể tham khảo ở đây

http://docs.ansible.com/ansible/list_of_all_modules.html . Được chia các module chính như: db, file, monitor, network, package, storage, web, cloud....

3. Ansible playbook

Ansible rất linh hoạt khi hỗ trợ playbook bằng ngôn ngữ YAML (file.yml). Ansible hỗ trợ với rất nhiều **module** giúp ta chạy các lệnh dễ dàng và giúp ta dễ đọc hiểu. playbook đơn giản có mẫu như sau:

```
//nano playbook.yml
---
- hosts: web_app
  become: yes
  tasks:
    - name: Ping host
      ping: ~
    - name: Install nginx
      apt:
        name: nginx
```

Các module được tùy biến trên nhiều hệ điều hành khác nhau. Ví dụ như trên Ubuntu để cài đặt dịch vụ ta dùng module **apt** còn trên CentOS ta dùng module **yum**.

```
//nano playbook.yml
---
- hosts: web_app
  become: yes
  tasks:
    - name: Ping host
      ping: ~
    - name: Install nginx
      yum:
        name: nginx
```

Để chạy playbook ta sử dụng lệnh sau:

```
ansible-playbook playbook.yml
```

```
ubuntu@ip-172-31-29-124:/etc/ansible$ ansible-playbook  playbook.yml

PLAY [web_app] *****
TASK [Gathering Facts] *****
ok: [172.31.13.222]
TASK [Ping host] *****
ok: [172.31.13.222]
TASK [Install nginx] *****
changed: [172.31.13.222]
PLAY RECAP *****
172.31.13.222      : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0
ubuntu@ip-172-31-29-124:/etc/ansible$
```

Kiểm tra xem nginx đã được cài trên web_app chưa:

```

ubuntu@ip-172-31-29-124:/etc/ansible$ curl 172.31.13.222
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
ubuntu@ip-172-31-29-124:/etc/ansible$ █

```

Dịch vụ nginx cài đặt thành công, playbook chạy thành công.

Tương tự ta có thể viết playbook để update máy server và copy file index lên server web_app. Tạo file với nội dung sau:

```

//nano playbook.yml
---
- hosts: web_app
  become: yes
  tasks:
    - name: Update and upgrade apt packages
      apt:
        upgrade: yes
        update_cache: yes
    - name: Copy file
      copy:
        src: index.html
        dest: /var/www/html/
    - name: Restart nginx
      service:
        name: nginx
        state: restarted

```

Tạo file index.html với nội dung bất kì lưu ở đường dẫn /etc/ansible/ và chạy playbook rồi kiểm tra kết quả:


```

ubuntu@ip-172-31-29-124:/etc/ansible$ ansible-playbook  playbook.yml

PLAY [web_app] *****

TASK [Gathering Facts] *****
ok: [172.31.13.222]

TASK [Update and upgrade apt packages] *****
ok: [172.31.13.222]

TASK [Copy file] *****
changed: [172.31.13.222]

TASK [Restart nginx] *****
changed: [172.31.13.222]

PLAY RECAP *****
172.31.13.222      : ok=4    changed=2    unreachable=0    failed=0

ubuntu@ip-172-31-29-124:/etc/ansible$ curl 172.31.13.222
<h1>Training DevOps 2022</h1>
ubuntu@ip-172-31-29-124:/etc/ansible$ █

```

Ta thấy nội dung file index đã được thay đổi. playbook chạy thành công.

Tương tự các module khác ta có thể search google.

- Thay vì viết module apt cho từng gói cài đặt ta có thể nhóm vào **item** để chạy 1 lần và sử dụng **handlers** để thực hiện các hành động cần thực thi nhiều lần (notify).

```

//playbook.yml
---
- hosts: web_app
  become: yes
  tasks:
    - name: Install Apache.
      apt:
        name: "{{ item }}"
        state: present
      with_items:
        - apache2
        - mysql-server
    - name: deploy html file
      template:
        src: /tmp/index.html
        dest: /var/www/html/index.html
      notify: restart web
  handlers:
    - name: restart web
      service:
        name: "{{ item }}"
        state: running
      with_items:

```

- apache2
- mysql

Run playbook và kiểm tra kết quả playbook chạy thành công.

```
ubuntu@ip-172-31-29-124:/etc/ansible$ ansible-playbook playbook.yml

PLAY [web_app] *****

TASK [Gathering Facts] *****
ok: [172.31.13.222]

TASK [Install Apache.] *****
ok: [172.31.13.222] => (item=apache2)
ok: [172.31.13.222] => (item=mysql-server)

TASK [deploy html file] *****
ok: [172.31.13.222]

PLAY RECAP *****
172.31.13.222 : ok=3    changed=0    unreachable=0    failed=0

ubuntu@ip-172-31-29-124:/etc/ansible$ curl 172.31.13.222
<h1>Training DevOps 2022</h1>
ubuntu@ip-172-31-29-124:/etc/ansible$
```

- Ta có thể khai báo biến trong playbook như sau:

```
//nano playbook.yml
---
- hosts: web_app
  become: yes
  vars:
    - index_file: "index.html"
    - variable: "Test var Ansible"
  tasks:
    - name: Install Apache.
      apt:
        name: "{{ item }}"
        state: present
      with_items:
        - apache2
        - mysql-server
    - name: deploy html file
      template:
        src: "{{ index_file }}"
        dest: /var/www/html/index.html
      notify: restart web
  handlers:
    - name: restart web
      service:
        name: "{{ item }}"
        state: running
      with_items:
```

- apache2
- mysql

Thay đổi nội dung file index.html như sau:

```
ubuntu@ip-172-31-29-124:/etc/ansible$ cat index.html
<h1>Training DevOps 2022</h1>
<h2>{{variable}}</h2>
ubuntu@ip-172-31-29-124:/etc/ansible$
```

Chạy và kiểm tra kết quả ta đã lấy được biến khai báo trong playbook.

```
ubuntu@ip-172-31-29-124:/etc/ansible$ ansible-playbook playbook.yml

PLAY [web_app] *****

TASK [Gathering Facts] *****
ok: [172.31.13.222]

TASK [Install Apache.] *****
ok: [172.31.13.222] => (item=apache2)
ok: [172.31.13.222] => (item=mysql-server)

TASK [deploy html file] *****
changed: [172.31.13.222]
```

```
ubuntu@ip-172-31-29-124:/etc/ansible$ curl 172.31.13.222
<h1>Training DevOps 2022</h1>
<h2>Test var Ansible</h2>
ubuntu@ip-172-31-29-124:/etc/ansible$
```

4. Role

Trong Ansible, **Role** là cơ chế tách 1 playbook ra thành nhiều file. Việc này nhằm đơn giản hóa việc viết các playbook phức tạp có thể tái sử dụng nhiều lần. Mỗi role là một thành phần độc lập, bao gồm nhiều **variables**, **tasks**, **files**, **templates** và **modules** bên dưới.

Để dễ hình dung ta sẽ làm lab với role **web** có **task** và **handlers** như sau:

```
ubuntu@ip-172-31-29-124:/etc/ansible$ tree
.
├── Training.pem
├── ansible.cfg
├── hosts
├── index.html
├── playbook.yml
├── roles
│   └── web
│       ├── handlers
│       │   └── main.yml
│       └── tasks
│           └── main.yml
└── 4 directories, 7 files
```

Nội dung các file như sau:

playbook.yml

```
---
- hosts: web_app
  become: yes
  tasks:
    - name: include web role
      include_role:
        name: web
      tags:
        - deployments
```

/roles/web/tasks/main.yml

```
---
- name: Install Apache.
  apt:
    name: "{{ item }}"
    state: present
  with_items:
    - apache2
    - mysql-server
- name: deploy html file
  template:
    src: index.html
    dest: /var/www/html/index.html
  notify: restart web
```

/roles/web/handlers/main.yml

```
---
- name: restart web
  service:
    name: "{{ item }}"
    state: restarted
  with_items:
    - apache2
    - mysql
```

Chạy playbook và kiểm tra kết quả play chạy thành công.

```
ubuntu@ip-172-31-29-124:/etc/ansible$ ansible-playbook playbook.yml

PLAY [web_app] *****

TASK [Gathering Facts] *****
ok: [172.31.13.222]

TASK [include web role] *****

TASK [web : Install Apache.] *****
ok: [172.31.13.222] => ( item=apache2)
ok: [172.31.13.222] => ( item=mysql-server)

TASK [web : deploy html file] *****
ok: [172.31.13.222]

PLAY RECAP *****
172.31.13.222      : ok=3    changed=0    unreachable=0    failed=0    skipped=0

ubuntu@ip-172-31-29-124:/etc/ansible$ curl 172.31.13.222
<h1>Training DevOps 2022</h1>
```

Thông thường các playbook có rất nhiều thành phần nên việc phân chia ra các role giúp việc quản lý dễ dàng hơn. Trên đây chỉ là ví dụ nhỏ để ta hình dung ra cấu trúc của roles.

```

ubuntu@ip-172-31-29-124:~/ansible-galaxy$ ansible-galaxy init demo
- Role demo was created successfully
ubuntu@ip-172-31-29-124:~/ansible-galaxy$ ll
total 12
drwxrwxr-x  3 ubuntu ubuntu 4096 Dec 11 21:37 ./
drwxr-xr-x 13 ubuntu ubuntu 4096 Dec 11 21:11 ../
drwxrwxr-x 10 ubuntu ubuntu 4096 Dec 11 21:37 demo/
ubuntu@ip-172-31-29-124:~/ansible-galaxy$ tree
.
├── demo
│   ├── README.md
│   ├── defaults
│   │   └── main.yml
│   ├── files
│   ├── handlers
│   │   └── main.yml
│   ├── meta
│   │   └── main.yml
│   ├── tasks
│   │   └── main.yml
│   ├── templates
│   ├── tests
│   │   ├── inventory
│   │   └── test.yml
│   └── vars
│       └── main.yml

```

- Remove role

```
ansible-galaxy remove username.role_name
```

```

ubuntu@ip-172-31-29-124:~$ ansible-galaxy remove geerlingguy.nginx
- successfully removed geerlingguy.nginx
ubuntu@ip-172-31-29-124:~$ █

```

Day 4, 5. Unit 3: Terraform

Assignment :

1. Install Terraform and AWS CLI

Để thực hành Terraform ta cần cài đặt AWS CLI và Terraform. Ta có thể cài đặt và sử dụng trên nhiều hệ điều hành nhưng trong bài thực hành này ta sẽ cài đặt và sử dụng trên Windows cho thuận tiện cho việc viết configure file trên visual code.

Để cài đặt Terraform trên Windows ta tải phần mềm trên trang chủ terraform.io:

https://releases.hashicorp.com/terraform/1.1.0/terraform_1.1.0_windows_amd64.zip

Để cài đặt AWS CLI ta tải bộ cài đặt MSI tại:

<https://awscli.amazonaws.com/AWSCLIV2.msi>

Sau khi cài đặt xong ta kiểm tra với lệnh sau:

```
C:\Users\ASUS GL552>aws --version
aws-cli/2.2.23 Python/3.8.8 Windows/10 exe/AMD64 prompt/off

C:\Users\ASUS GL552>_
```

Tiếp theo ta thêm access key vào bằng lệnh sau:

```
C:\Users\ASUS GL552>aws --version
aws-cli/2.2.23 Python/3.8.8 Windows/10 exe/AMD64 prompt/off

C:\Users\ASUS GL552>aws configure
AWS Access Key ID [*****3CGE]:
AWS Secret Access Key [*****jhKv]:
Default region name [ap-east-1]:
Default output format [json]: _
```

Làm tương tự như module AWS.

Ta cũng có thể tham khảo cách cài đặt Terraform và AWS CLI tại:

<https://cloudlinuxtech.com/install-terraform-on-ubuntu-uninstall-terraform/>

https://linuxhint.com/install_aws_cli_ubuntu/

2. Run the first time

Terraform hỗ trợ rất nhiều provider khác nhau tuy nhiên trong bài lab này ta tập trung tìm hiểu về provider aws để tạo infrastructure trên aws.

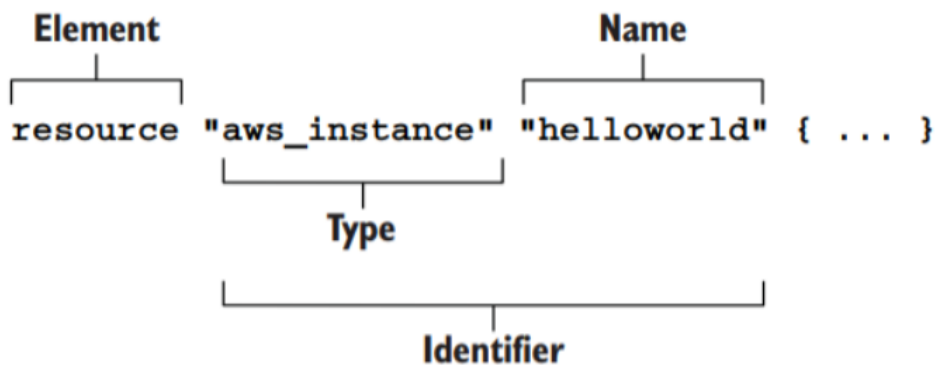
Tạo thư mục và mở bằng visual studio code. Khai báo file đầu tiên main.tf với nội dung:

```
provider "aws" {  
  region = "ap-east-1"  
}
```

Ở đây chúng ta khai báo provider là aws và vùng HongKong. Sau đó ta khai báo instance bằng đoạn code sau:

```
resource "aws_instance" "training" {  
  ami      = "ami-0b215afe809665ae5"  
  instance_type = "t3.micro"  
  tags = {  
    Name = "training"  
  }  
}
```





Ở trên ta sử dụng một block tên là resources, đây là block quan trọng nhất của terraform, ta sẽ sử dụng block này để tạo resource của chúng ta. Phía sau resources thì ta sẽ có thêm giá trị nữa là resource type mà ta muốn tạo (cái này phụ thuộc vào provider của chúng ta sẽ cung cấp những resource type nào) , ở trên resource type của ta là **aws_instance**, và giá trị cuối cùng là tên của resource đó, này ta muốn đặt gì cũng được.



Để xem những thuộc tính của một resource nào đó, ta lên trang <https://registry.terraform.io/> để xem.

Trong đoạn code trên ta cần chú ý ami là loại instance ta muốn cài đặt nó phân biệt theo region nên ta có thể vào **launch instance** để tìm kiếm và instance_type ta chọn loại **free tier eligible** như trong module AWS.

Amazon Machine Image (AMI)

	Amazon Linux 2 AMI (HVM) - Kernel 4.14, SSD Volume Type - ami-0a7fb2d401b6017e5 (64-bit x86) / ami-07fe2c4d7640f1b2d (64-bit Arm)
Amazon Linux Free tier eligible	Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is now under maintenance only mode and has been removed from this wizard.
	Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
	Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-0ec87970e52e19867 (64-bit x86) / ami-047f2232912795fcf (64-bit Arm)
Red Hat Free tier eligible	Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type
	Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
	SUSE Linux Enterprise Server 15 SP2 (HVM), SSD Volume Type - ami-0b4017973f2328b15 (64-bit x86) / ami-0748db038a2205f21 (64-bit Arm)
SUSE Linux Free tier eligible	SUSE Linux Enterprise Server 15 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.5, PHP 5 and Ruby 1.8.7 available.
	Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
	Ubuntu Server 20.04 LTS (HVM), SSD Volume Type - ami-0a9c1cc3697104990 (64-bit x86) / ami-0b112a0b6eb576c48 (64-bit Arm)
Ubuntu Free tier eligible	Ubuntu Server 20.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).
	Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Xong khi ta viết config xong hết, thì ta mở terminal lên và gõ `terraform init`, bước này là bắt buộc khi ta viết một cấu hình cho một hạ tầng mới, nó sẽ tải code của provider xuống folder hiện tại mà ta viết file `main.tf`.

```
PS D:\Terraform\Test> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v3.69.0...
- Installed hashicorp/aws v3.69.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS D:\Terraform\Test>
```

Sau khi init xong, ta gõ tiếp câu lệnh apply để nó tạo EC2 cho ta.

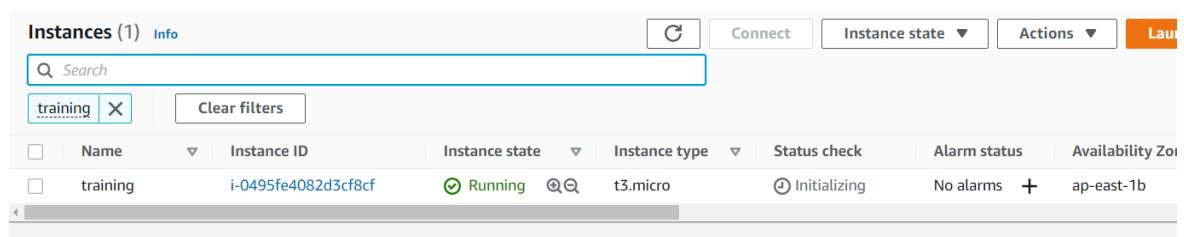
```
PS D:\Terraform\Test> terraform apply

Terraform used the selected providers to generate the following execution plan. Resources to be
+ create

Terraform will perform the following actions:

# aws_instance.training will be created
+ resource "aws_instance" "training" {
  + ami                        = "ami-0b215afe809665ae5"
  + arn                       = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone          = (known after apply)
  + cpu_core_count             = (known after apply)
  + cpu_threads_per_core       = (known after apply)
  + disable_api_termination    = (known after apply)
  + ebs_optimized              = (known after apply)
  + get_password_data          = false
  + host_id                    = (known after apply)
  + id                         = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
}
```

Ta đọc cấu hình kiểm tra các thông tin và enter “yes” để terraform tạo instance.



Bây giờ nếu ta muốn xóa EC2 đi, ta chỉ cần chạy câu lệnh destroy.

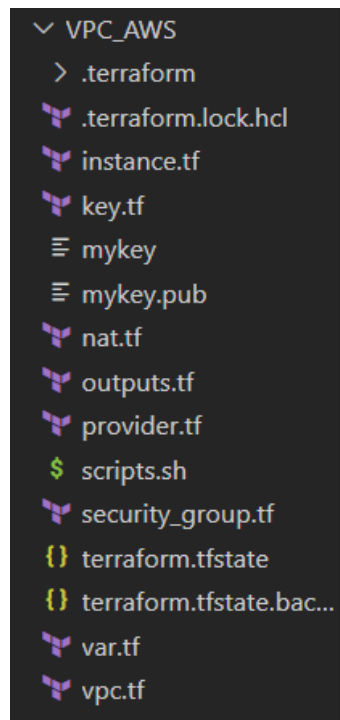
```
PS D:\Terraform\Test> terraform destroy --auto-approve
aws_instance.training: Refreshing state... [id=i-0495fe4082d3cf8cf]

Terraform used the selected providers to generate the following execution plan. Resources to be
- destroy

Terraform will perform the following actions:

# aws_instance.training will be destroyed
- resource "aws_instance" "training" {
  - ami                        = "ami-0b215afe809665ae5" -> null
  - arn                       = "arn:aws:ec2:ap-east-1:179623033511:ins
  - associate_public_ip_address = true -> null
  - availability_zone          = "ap-east-1b" -> null
  - cpu_core_count             = 1 -> null
  - cpu_threads_per_core       = 2 -> null
  - disable_api_termination    = false -> null
}
```

Để dễ dàng quản lý các resource ta có thể phân chia ra thành các file nhỏ như sau:



3. AWS resource

Một số resource AWS cơ bản

- aws_instance

```
resource "aws_instance" "name" {  
    ami                = "ami-"  
    instance_type      = "t3.micro"  
    subnet_id          = aws_subnet.  
    security_groups    = [aws_security_group.]  
    key_name            = aws_key_pair.  
    ebs_block_device {device_name, volume_size}  
    tags               = { Name}  
    provisioner "name" {}  
    connection {host, type, user, private_key}  
}
```

aws_instance.name.id

- variable

variable "name" {type, default}

var.name

- aws_key_pair

```
resource "aws_key_pair" "name" {key_name, public_key}
```

aws_key_pair.name.key_name

- output

output "name" {value}

- aws_vpc

```
resource "aws_vpc" "name" {  
  cidr_block      = "192.168.0.0/16"  
  instance_tenancy = "default"  
  enable_dns_support = "true"  
  enable_dns_hostnames = "true"  
  enable_classiclink = "false"  
  tags = {Name}  
}
```

aws_vpc.name.id

- aws_subnet

```
resource "aws_subnet" "name" {  
  vpc_id      = aws_vpc.name.id  
  cidr_block   = "192.168.1.0/24"  
  map_public_ip_on_launch = "true/false"  
  availability_zone = "ap-east-1a"  
  tags = { Name}  
}
```

aws_subnet.name.id

- aws_internet_gateway

```
resource "aws_internet_gateway" "name" {  
  vpc_id = aws_vpc.name.id  
  tags = { Name}  
}
```

aws_internet_gateway.name.id

- aws_route_table

```
resource "aws_route_table" "name" {  
  vpc_id = aws_vpc.name.id  
  route {  
    cidr_block = "0.0.0.0/0"  
    gateway_id = aws_internet_gateway.name.id  
  }  
  tags = { Name}  
}
```

aws_route_table.name.id

- aws_route_table_association

```
resource "aws_route_table_association" "name" {
  subnet_id      = aws_subnet.name.id
  route_table_id = aws_route_table.name.id
}
```

aws_route_table_association.name.id

- aws_nat_gateway

```
resource "aws_eip" "nat" {
  vpc = true
}
resource "aws_nat_gateway" "nat_gateway" {
  allocation_id = aws_eip.nat.id
  subnet_id     = aws_subnet.public_subnet.id
  depends_on    = [aws_internet_gateway.name]
}
```

- aws_security_group

```
resource "aws_security_group" "name" {
  vpc_id      = aws_vpc.name.id
  name        = "allow-all"
  egress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {Name}
}
```

aws_security_group.name.id

- data

data "type" "name" {}

data.type.name.

- module

module "name" {}

module.name.

- aws_ebs_volume

```

resource "aws_ebs_volume" "name" {
  availability_zone = "ap-east-1a"
  size             = 20
  type             = "gp2"
  tags = { Name}
}
resource "aws_volume_attachment" "name" {
  device_name = "/dev/xvdh"
  volume_id   = aws_ebs_volume.name.id
  instance_id = aws_instance.name.id
}

```

4. Create infrastructure on AWS

Viết configure file tạo hạ tầng trên AWS bao gồm: VPC, public subnet, private subnet, NAT, Internet Gateway, Route Table, Security Group, Instance.

```

//provider.tf
provider "aws" {
  profile = "default"
  region  = var.AWS_REGION
}

```

```

//vcp.tf
resource "aws_vpc" "VLAN" {
  cidr_block           = "192.168.0.0/16"
  instance_tenancy     = "default"
  enable_dns_support   = "true"
  enable_dns_hostnames = "true"
  enable_classiclink   = "false"
  tags = {
    Name = "ThuongDD"
    Environment = var.ENV
  }
}

resource "aws_subnet" "public_subnet" {
  vpc_id            = aws_vpc.VLAN.id
  cidr_block        = "192.168.0.0/24"
  map_public_ip_on_launch = "true"
  availability_zone  = "ap-east-1a"
  tags = {
    Name = "public_subnet"
    Environment = var.ENV
  }
}

resource "aws_subnet" "public_subnet_1" {
  vpc_id            = aws_vpc.VLAN.id

```

```

    cidr_block          = "192.168.1.0/24"
    map_public_ip_on_launch = "true"
    availability_zone     = "ap-east-1b"
    tags = {
        Name = "public_subnet_1"
        Environment = var.ENV
    }
}
resource "aws_subnet" "private_subnet" {
    vpc_id          = aws_vpc.VLAN.id
    cidr_block      = "192.168.2.0/24"
    map_public_ip_on_launch = "false"
    availability_zone = "ap-east-1a"
    tags = {
        Name = "private_subnet"
        Environment = var.ENV
    }
}
resource "aws_subnet" "private_subnet_1" {
    vpc_id          = aws_vpc.VLAN.id
    cidr_block      = "192.168.3.0/24"
    map_public_ip_on_launch = "false"
    availability_zone = "ap-east-1b"
    tags = {
        Name = "private_subnet_1"
        Environment = var.ENV
    }
}
resource "aws_internet_gateway" "internet_gateway" {
    vpc_id = aws_vpc.VLAN.id
    tags = {
        Name = "internet_gateway_thuongdd"
        Environment = var.ENV
    }
}
resource "aws_route_table" "route_table" {
    vpc_id = aws_vpc.VLAN.id
    route {
        cidr_block = "0.0.0.0/0"
        gateway_id = aws_internet_gateway.internet_gateway.id
    }
    tags = {
        Name = "route_table_thuongdd"
        Environment = var.ENV
    }
}
resource "aws_route_table_association" "route_table_association" {
    subnet_id      = aws_subnet.public_subnet.id
    route_table_id = aws_route_table.route_table.id
}

```

```

}
resource "aws_route_table_association" "route_table_association_1" {
  subnet_id      = aws_subnet.public_subnet_1.id
  route_table_id = aws_route_table.route_table.id
}

```

```

//security_group.tf
resource "aws_security_group" "public" {
  vpc_id = aws_vpc.VLAN.id
  name   = "allow-all"
  egress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name = "Security"
  }
}
resource "aws_security_group" "private" {
  vpc_id = aws_vpc.VLAN.id
  name   = "allow_bastion_host"
  egress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name       = "Security"
    Environment = var.ENV
  }
}

```



```

//nat.tf
resource "aws_eip" "nat" {
  vpc = true
}
resource "aws_nat_gateway" "nat_gateway" {
  allocation_id = aws_eip.nat.id
  subnet_id     = aws_subnet.public_subnet.id
  depends_on    = [aws_internet_gateway.internet_gateway]
}
resource "aws_route_table" "nat" {
  vpc_id = aws_vpc.VLAN.id
  route {
    cidr_block      = "0.0.0.0/0"
    nat_gateway_id = aws_nat_gateway.nat_gateway.id
  }
  tags = {
    Name = "nat_private"
    Environment = var.ENV
  }
}
resource "aws_route_table_association" "route_table_association_2" {
  subnet_id      = aws_subnet.private_subnet.id
  route_table_id = aws_route_table.nat.id
}
resource "aws_route_table_association" "route_table_association_3" {
  subnet_id      = aws_subnet.private_subnet_1.id
  route_table_id = aws_route_table.nat.id
}

```

```

//instance.tf
resource "aws_instance" "node_1" {
  ami          = "ami-0b215afe809665ae5"
  instance_type = "t3.small"
  subnet_id    = aws_subnet.public_subnet.id
  security_groups = [aws_security_group.private.id]
  key_name     = var.key_name
  tags = {
    Name = "node_1"
    Environment = var.ENV
  }
}
resource "aws_instance" "node_2" {
  ami          = "ami-0b215afe809665ae5"
  instance_type = "t3.small"
  subnet_id    = aws_subnet.private_subnet.id
  security_groups = [aws_security_group.private.id]
  key_name     = var.key_name
}

```

```
tags = {
  Name = "node_2"
}
```

```
//output.tf
output "bastion_host" {
  value = aws_instance.bastion_host.public_ip
}
output "private_master" {
  value = aws_instance.master.private_ip
}
output "private_node_1" {
  value = aws_instance.node_1.private_ip
}
output "private_node_2" {
  value = aws_instance.node_2.private_ip
}
```

```
//var.tf
variable "AWS_REGION" {
  default = "ap-east-1"
}

variable "instance_username" {
  default = "ubuntu"
}

variable "ENV" {
  default = "test"
}
```

Objectives:

- know Terraform and create infrastructure on AWS

Problem Descriptions:

Assumptions:

Technical Requirements:

- AWS Infrastructure

Questions to answer:

Estimated Time to complete: 700 mins

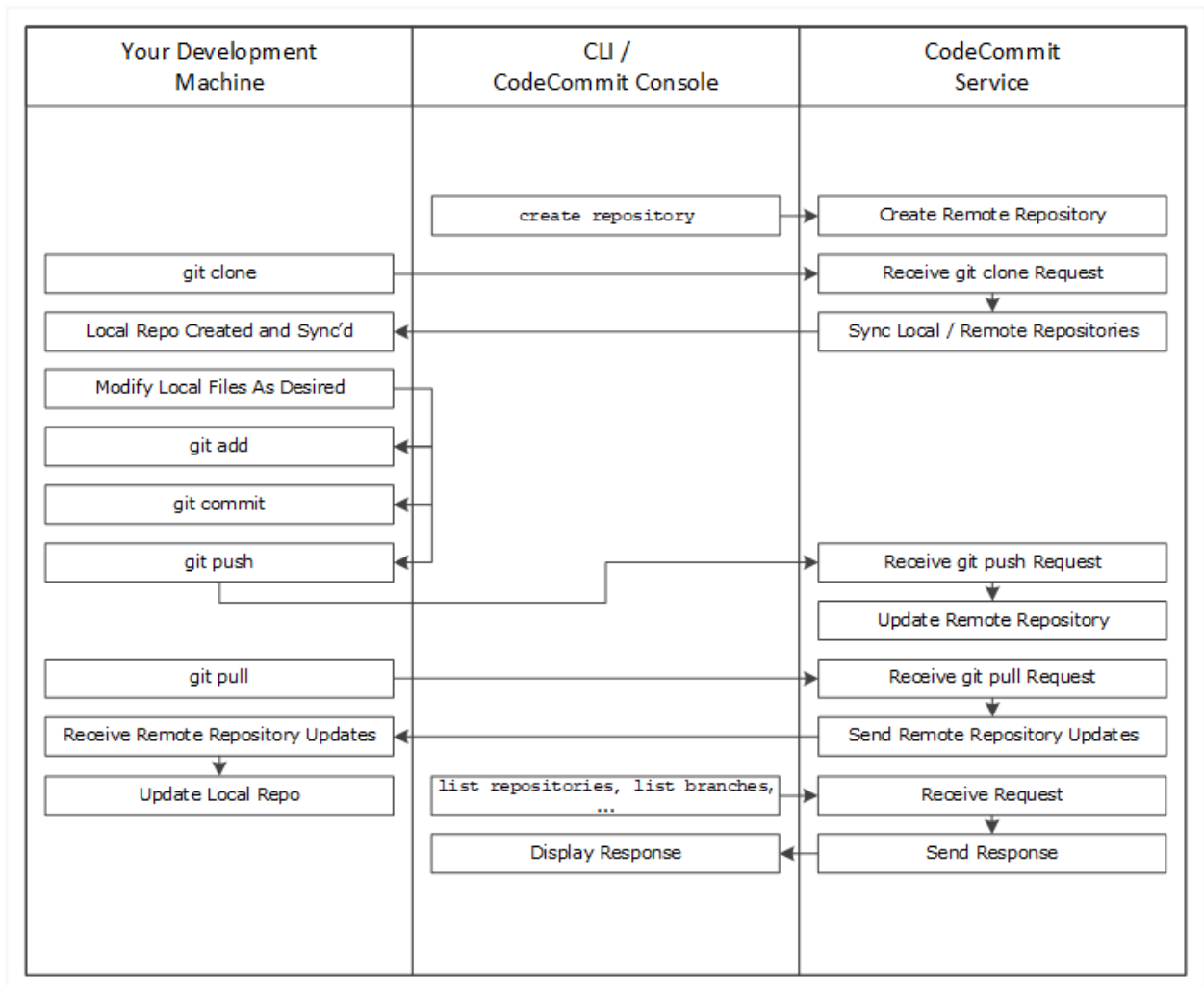
Day 6, 7. Unit 4: CodePipeline

Assignment 1:

CodePipeline

1. CodeCommit

CodeCommit là một service dùng để quản lý, lưu trữ các phiên bản source code dự án của chúng ta tương tự như Github một cách bảo mật, an toàn, độ khả dụng cao và khả năng mở rộng với các service mà chúng ta đang sử dụng trên AWS.



Đăng ký CodeCommit Service để lưu trữ source code:

Ta vào AWS console tìm kiếm **CodeCommit** chọn **Create repository**

Create repository

Create a secure repository to store and share your code. Begin by typing a repository name and a description for your repository. Repository names are included in the URLs for that repository.

Repository settings

Repository name

100 characters maximum. Other limits apply.

Description - *optional*

1,000 characters maximum

Sau khi đăng ký kho lưu trữ trên AWS thì chúng ta **config SSH** để push code lên. Ta sẽ gen key tại máy localhost bằng lệnh:

ssh-keygen

Sau đó ta copy nội dung file .pub là public key lên AWS. Ta vào **AWS console -> IAM -> Access Management -> Users -> Security Credentials** và thêm nội dung file .pub và như hình.

Upload SSH public key

SSH key ID

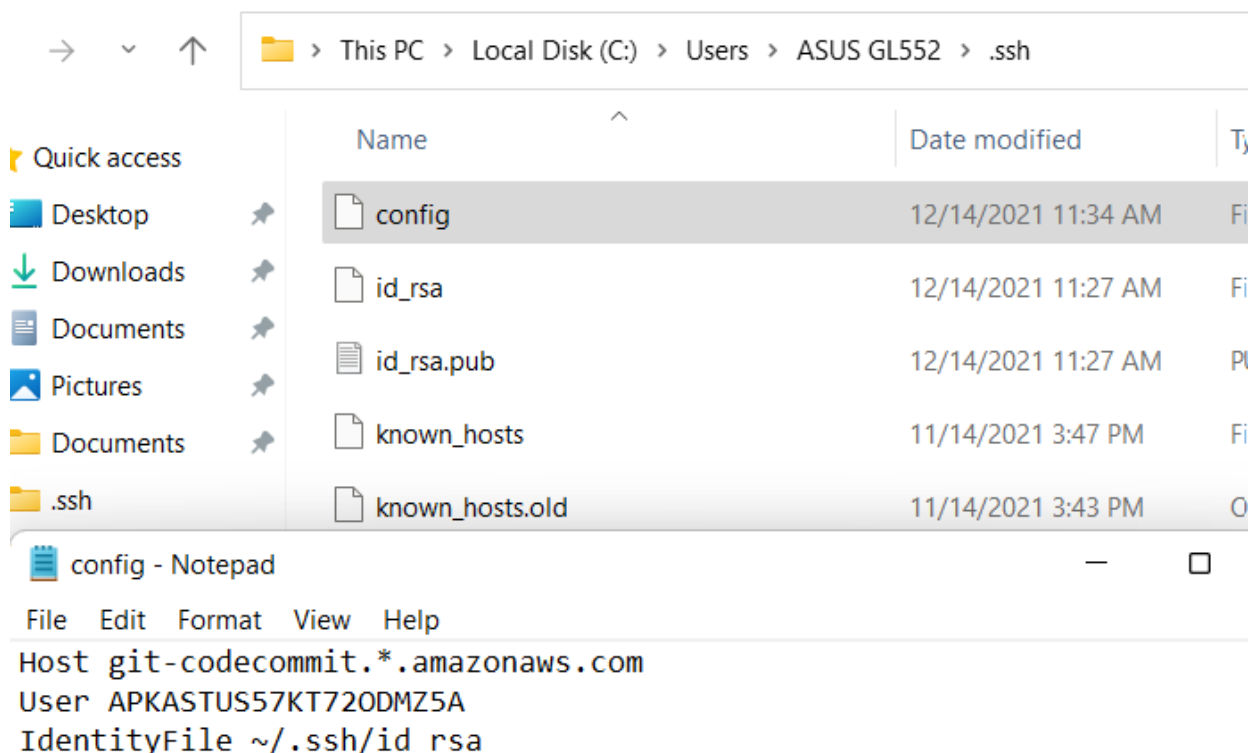
APKASTUS57KT72ODMZ5A | [Show SSH key](#)

HTTPS Git credentials for AWS CodeCommit

Generate a user name and password you can use to authenticate HTTPS connections to AWS CodeC
[more](#)

Lấy IAM-SSH-Key-ID đã tạo ở trên đưa vào config trong ~/.ssh/ với nội dung:

```
Host git-codecommit.*.amazonaws.com
  User IAM-SSH-Key-ID
  IdentityFile ~/.ssh/id-rsa
```



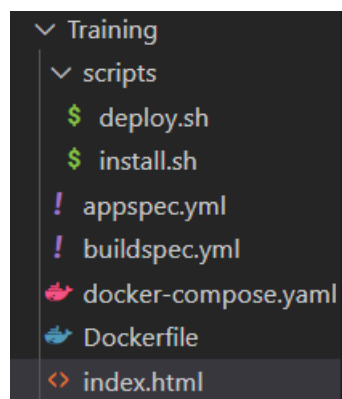
Tiếp theo ta tạo repository ở máy local và push lên remote repository tương tự như GitHub.

```
git remote add origin [repository]
git branch -M master
git add .

git commit -m "init project"

git push origin master
```

Nội dung source code như sau:



```
//index.html
<h1>Training DevOps 2022</h1>
```

```
//Dockerfile
FROM nginx:1.13.9-alpine
COPY index.html /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

```
//docker-compose.yaml
version: "3.3"
services:
  app:
    image: 179623033511.dkr.ecr.ap-east-1.amazonaws.com/training:latest
    ports:
      - "80:80"
```

Chỉnh sửa image theo url ECR.

```
//buildspec.yml
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws ecr get-login-password --region ap-east-1 | docker login --username AWS
--password-stdin 179623033511.dkr.ecr.ap-east-1.amazonaws.com
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t training .
      - docker tag training:latest 179623033511.dkr.ecr.ap-east-
1.amazonaws.com/training:latest
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push 179623033511.dkr.ecr.ap-east-1.amazonaws.com/training:latest
artifacts:
  files: appspec.yml
```

Chỉnh sửa url ECR theo ECR đã tạo.

```
//appspec.yml
version: 0.0
os: linux
files:
  - source: /docker-compose.yaml
    destination: /home/ubuntu
hooks:
  Install:
    - location: scripts/install.sh
      timeout: 300
      runas: ubuntu
  ApplicationStart:
    - location: scripts/deploy.sh
      timeout: 300
```

```
//scripts/install.sh
#!/bin/bash
sudo apt update
sudo apt install docker -y
sudo apt install docker-compose -y
sudo apt install awscli -y
sudo aws ecr get-login-password --region ap-east-1 | docker login --username AWS --
password-stdin 179623033511.dkr.ecr.ap-east-1.amazonaws.com
```

Chỉnh sửa url ECR cho phù hợp.

```
//scripts/deploy.sh
#!/bin/bash
cd /home/ubuntu/
docker-compose stop
docker-compose rm -f
docker-compose pull
docker-compose up -d
```

2. Elastic Container Registry

Amazon Elastic Container Registry (**ECR**) là dịch vụ lưu trữ Docker images được quản lý đầy đủ giúp các nhà phát triển dễ dàng lưu trữ, quản lý và triển khai Docker images.

Trong bài lab này ta sẽ sử dụng ECR để sau khi build xong ta sẽ lưu images vào ECR nên ta cần tạo ECR và role cho CodeBuild truy cập được ECR.

Để tạo ECR ta vào AWS console tìm kiếm ECR -> Create repository.

Sau khi tạo xong repository ta mở **Views push commands** để chỉnh sửa source link dẫn đến repository cho phù hợp.

Push commands for training

```
aws ecr get-login-password --region ap-east-1 | docker login --username AWS --password-stdin
```

Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.

2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions [here](#). You can skip this step if your image is already built:

```
docker build -t training .
```

3. After the build completes, tag your image so you can push the image to this repository:

```
docker tag training:latest 179623033511.dkr.ecr.ap-east-1.amazonaws.com/training:latest
```

4. Run the following command to push this image to your newly created AWS repository:

```
docker push 179623033511.dkr.ecr.ap-east-1.amazonaws.com/training:latest
```


3. CodeBuild

CodeBuild là dịch vụ AWS hỗ trợ quá trình tích hợp các thay đổi từ source code vào hệ thống một cách liên tục thông qua các hoạt động biên dịch, kiểm thử và đóng gói sản phẩm phần mềm để chuẩn bị cho giai đoạn triển khai kế tiếp.

Để tạo CodeBuild ta vào AWS console tìm kiếm **CodeBuild** -> **Build Project** -> **Create Build Project** và làm theo hướng dẫn:

[Developer Tools](#) > [CodeBuild](#) > [Build projects](#) > [Create build project](#)

Create build project

Project configuration

Project name

A project name must be 2 to 255 characters. It can include the letters A-Z and a-z, the numbers 0-9, and

Description - *optional*

Build badge - *optional*

☐ Enable build badge

Enable concurrent build limit - *optional*

Limit the number of allowed concurrent builds for this project.

☐ Restrict number of concurrent builds this project can start

► Additional configuration

tags

Source 1 - Primary

Source provider

AWS CodeCommit

Repository

Training

Reference type

Choose the source version reference type that contains your source code.

- ☒ Branch
- ☐ Git tag
- ☐ Commit ID

Branch

Choose a branch that contains the code to build.

master

Commit ID - *optional*

Choose a commit ID. This can shorten

2f271a4378fcb01e4a2b6

Environment


Environment image

☒ Managed image
Use an image managed by AWS CodeBuild

☐ Custom image
Specify a Docker image

Operating system

Ubuntu

 The programming language runtimes are now included in the standard image of Ubuntu 18.04, which is recommended for new CodeBuild projects created in the console. See [Docker Images Provided by CodeBuild for details](#).

Runtime(s)

Standard

Image

aws/codebuild/standard:5.0

Image version

aws/codebuild/standard:5.0-21.10.15

Privileged

- ☒ Enable this flag if you want to build Docker images or want your builds to get elevated privileges

Service role

☒ **New service role**
Create a service role in your account

☐ **Existing service role**
Choose an existing service role from your account

Role name

Type your service role name

► **Additional configuration**
Timeout, certificate, VPC, compute type, environment variables, file systems

Buildspec

Build specifications

☒ **Use a buildspec file**
Store build commands in a YAML-formatted buildspec file

☐ **Insert build commands**
Store build commands as build project configuration

Buildspec name - optional
By default, CodeBuild looks for a file named buildspec.yml in the source code root directory. If your buildspec file uses a different name or location, enter its path from the source root here (for example, buildspec-two.yml or configuration/buildspec.yml).

Lưu lại cấu hình và build ta thấy có lỗi liên quan đến login vào ECR nên ta cần thêm policy AmazonEC2ContainerRegistryPowerUser vào role đã tạo trong cấu hình codebuild.

Add permissions to codebuild-training-service-role

Attach Permissions

Create policy

Filter policies ▼

	Policy name ▼
<input type="checkbox"/>	► AmazonCognitoPowerUser
<input checked="" type="checkbox"/>	► AmazonEC2ContainerRegistryPowerUser
<input type="checkbox"/>	► AmazonElasticContainerRegistryPublicPowerUser
<input type="checkbox"/>	► AWSCodeCommitPowerUser
<input type="checkbox"/>	► AWSDataPipeline_PowerUser
<input type="checkbox"/>	► AWSKeyManagementServicePowerUser
<input type="checkbox"/>	► PowerUserAccess

Chúng ta build lại và kiểm tra kết quả CodeBuild chạy thành công.

Build logs	Phase details	Reports	Environment variables	Bu
Name	Status	Context	Duration	
SUBMITTED	✔ Succeeded	-	<1 sec	
QUEUED	✔ Succeeded	-	1 sec	
PROVISIONING	✔ Succeeded	-	180 secs	
DOWNLOAD_SOURCE	✔ Succeeded	-	4 secs	
INSTALL	✔ Succeeded	-	<1 sec	
PRE_BUILD	✔ Succeeded	-	1 sec	
BUILD	✔ Succeeded	-	6 secs	
POST_BUILD	✔ Succeeded	-	1 sec	
UPLOAD_ARTIFACTS	✔ Succeeded	-	<1 sec	
FINALIZING	✔ Succeeded	-	2 secs	
COMPLETED	✔ Succeeded	-	-	

Kiểm tra ECR thấy có image mới được đẩy lên:

Amazon ECR > Repositories > training

training

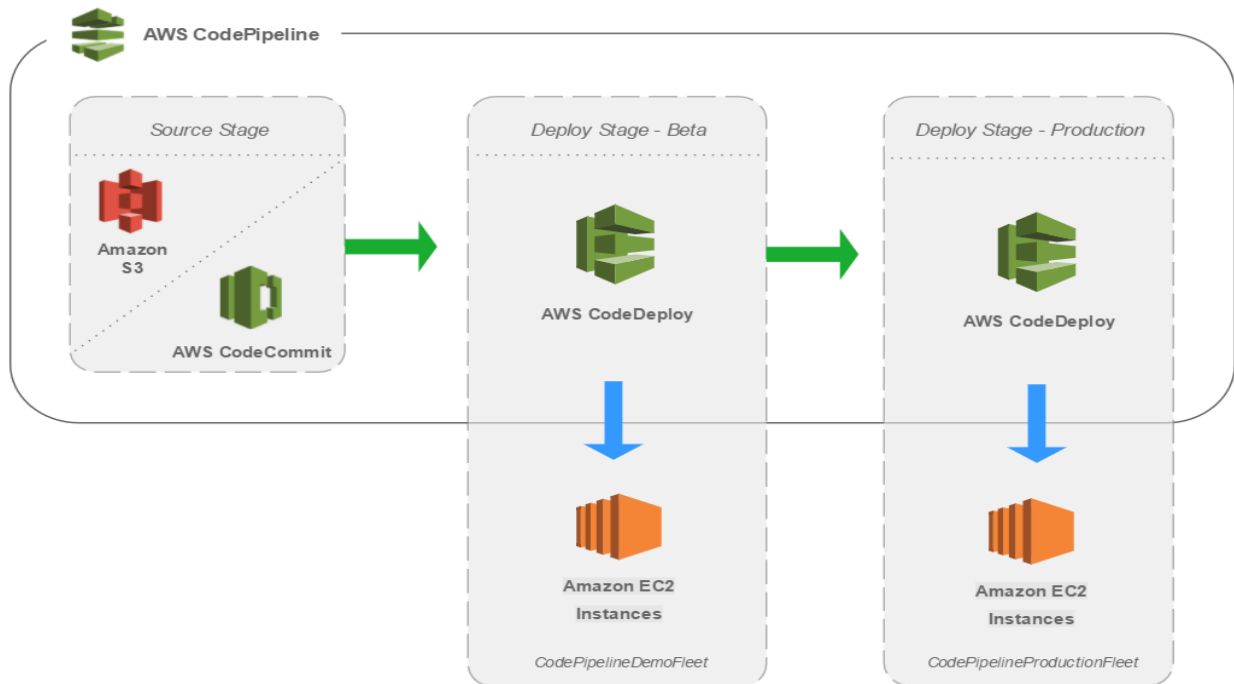
Images (7)

🔍 Find images

<input type="checkbox"/>	Image tag	Pushed at ▼	Size (MB) ▼	Image URI	Digest
<input type="checkbox"/>	latest	December 14, 2021, 13:47:45 (UTC+07)	7.87	Copy URI	sha256:ff1a6

4. CodeDeploy

AWS CodeDeploy là dịch vụ triển khai được quản lý toàn phần có khả năng tự động hóa việc triển khai phần mềm trên các dịch vụ điện toán như Amazon EC2, AWS Fargate, AWS Lambda và các máy chủ chạy tại chỗ của bạn. ... Dịch vụ thay đổi quy mô để phù hợp với nhu cầu triển khai của bạn.



Để CodeDeploy có thể chạy trên AWS instance ta cần IAM role cho phép instance có quyền trên s3 bucket để tải về và cài đặt **codedeploy agent**.

Cấu hình policy với nội dung sau:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Summary

Policy ARN `arn:aws:iam::179623033511`

Description

Permissions Policy usage Tags Policy versions Access

Policy summary {} JSON Edit policy

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Action": [
6         "s3:Get*",
7         "s3:List*"
8       ],
9       "Effect": "Allow",
10      "Resource": "*"
11    }
12  ]
13 }

```

Tạo role và attach policy vừa tạo vào như sau:

Instance Profile ARNs `arn:aws:iam::179623033511:instance-profile/CodeDeploy-Training-EC2`

Path /

Creation time 2021-12-14 14:41 UTC+0700

Last activity Not accessed in the tracking period

Maximum session duration 1 hour [Edit](#)

Permissions Trust relationships Tags Access Advisor Revoke sessions

▼ Permissions policies (1 policy applied)

[Attach policies](#)

Policy name ▼	Policy type ▼
▶ CodeDeploy-Training	Managed policy

Tạo instance ubuntu với tag Name hoặc Environment theo nhóm để deploy app. instance cần IAM role là IAM vừa tạo để có thể cài đặt được CodeDeploy agent.

Cài đặt CodeDeploy Agent trên instance:

```

sudo apt update
sudo apt install ruby-full

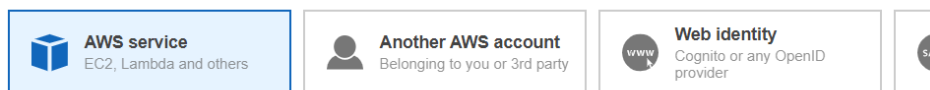
```

```
sudo apt install wget
wget https://bucket-name.s3.region-identifier.amazonaws.com/latest/install
ví dụ:
https:// aws-codedeploy-ap-east-1.s3.ap-east-1.amazonaws.com/latest/install
chmod +x ./install
sudo ./install auto > /tmp/logfile
sudo service codedeploy-agent status
sudo service codedeploy-agent start
```

Tham khảo tại:

<https://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install-ubuntu.html>

Tạo Role cho CodeDeploy ta vào **IAM -> Role -> Create role**



Choose a use case

Common use cases

EC2

Allows EC2 instances to call AWS services on your behalf.

Lambda

Allows Lambda functions to call AWS services on your behalf.

Or select a service to view its use cases

API Gateway	CloudWatch Events	EMR	IoT SiteWise
AWS Backup	CodeBuild	EMR Containers	IoT Things Graph
AWS Chatbot	CodeDeploy	ElastiCache	KMS

Chọn **CodeDeploy**

Select your use case

CodeDeploy

Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf.

CodeDeploy - ECS

Allows CodeDeploy to read S3 objects, invoke Lambda functions, publish to SNS topics, and update ECS s

CodeDeploy for Lambda

Allows CodeDeploy to route traffic to a new version of an AWS Lambda function version on your behalf.

* **Required**

Điền role name

Create role

Review

Provide the required information below and review this role before you create it.

Role name*

CodeDeploy-Training

Use alphanumeric and '+=,._@-' characters. Maximum 64 characters.

Role description

Allows CodeDeploy to call AWS services such as Auto Scaling

Maximum 1000 characters. Use alphanumeric and '+=,._@-' characters.

Trusted entities

AWS service: codedeploy.amazonaws.com

Policies



[AWSCodeDeployRole](#)

Permissions boundary

Permissions boundary is not set

Ta vào AWS console tìm kiếm **CodeDeploy** -> **Application** -> **Create application**

Developer Tools > CodeDeploy > Applications > Create application

Create application

Application configuration

Application name

Enter an application name

Training

100 character limit

Compute platform

Choose a compute platform

EC2/On-premises

Trong application ta **Create deployment group**

Service role

Enter a service role

Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.

arn:aws:iam::179623033511:role/CodeDeploy-Training



Deployment type

Choose how to deploy your application

☒ In-place

Updates the instances in the deployment group with the latest application revisions. During a deployment, each instance will be briefly taken offline for its update

☐ Blue/green

Replaces the instances in the deployment group with new instances and deploys the latest application revision to them. After instances in the replacement environment are registered with a load balancer, instances from the original environment are deregistered and can be terminated.

Environment configuration

Select any combination of Amazon EC2 Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add to this deployment

☐ Amazon EC2 Auto Scaling groups

☒ Amazon EC2 instances

1 unique matched instance. [Click here for details](#)

You can add up to three groups of tags for EC2 instances to this deployment group.

One tag group: Any instance identified by the tag group will be deployed to.

Multiple tag groups: Only instances identified by all the tag groups will be deployed to.

Tag group 1

Key

Value - optional

Name



training



Remove tag

Add tag

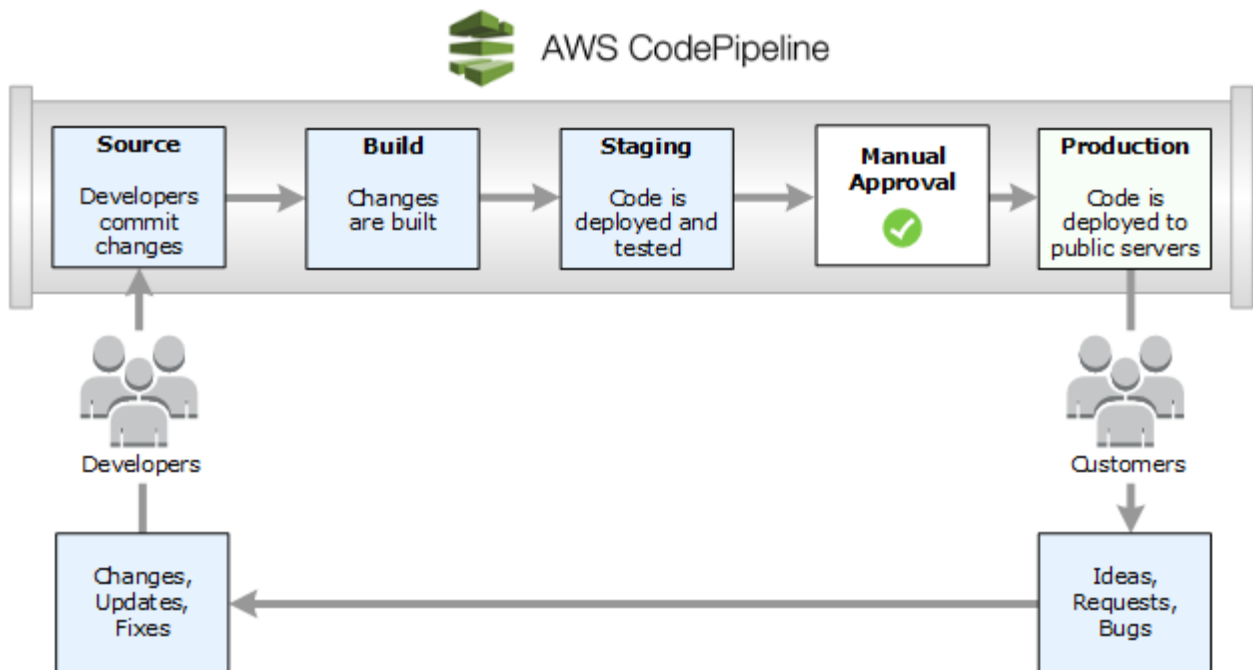
Add tag group

Đặt tag group theo instance ta đã tạo trước đó.

Phần này demo nên ta bỏ load balancer để đỡ tốn phí. Lưu cấu hình lại.

5. CodePipeline

CodePipeline là dịch vụ AWS cho phép chúng ta xây dựng quy trình triển khai ứng dụng một cách liên tục và tự động. Với cách thức cấu hình đơn giản, CodePipeline có khả năng mô hình hoá trực quan các bước cần thiết để biên dịch, kiểm thử và triển khai các phiên bản cập nhật cho một ứng dụng hoặc dịch vụ.



Để tạo Codepipeline ta vào AWS console tìm kiếm **Codepipeline** -> **Create pipeline** và làm theo hướng dẫn:

Pipeline settings

Pipeline name
Enter the pipeline name. You cannot edit the pipeline name after it is created.

No more than 100 characters

Service role

☒ **New service role**
Create a service role in your account

☐ **Existing service role**
Choose an existing service role from your account

Role name

Type your service role name

☒ **Allow AWS CodePipeline to create a service role so it can be used with this new pipeline**

Source

Source provider

This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

AWS CodeCommit

Repository name

Choose a repository that you have already created where you have pushed your source code.

Training

Branch name

Choose a branch of the repository

master

Change detection options

Choose a detection mode to automatically start your pipeline when a change occurs in the source code.

☒ **Amazon CloudWatch Events (recommended)**
Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs

☐ **AWS CodePipeline**
Use AWS CodePipeline to check periodically for changes

Add build stage [Info](#)

Build - optional

Build provider

This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.

AWS CodeBuild

Region

Asia Pacific (Hong Kong)

Project name

Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

Training

or

Create project

Environment variables - optional

Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

Add environment variable

Build type

☒ Single build

☐ Batch build

Add deploy stage [Info](#)

Deploy - optional

Deploy provider

Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

AWS CodeDeploy

Region

Asia Pacific (Hong Kong)

Application name

Choose an application that you have already created in the AWS CodeDeploy console. Or create an application in the AWS CodeDeploy console and then return to this task.

Training

Deployment group

Choose a deployment group that you have already created in the AWS CodeDeploy console. Or create a deployment group in the AWS CodeDeploy console and then return to this task.

Training

Lưu lại cấu hình và chờ pipeline chạy để kiểm tra kết quả:

Developer Tools

CodePipeline

► Source • CodeCommit

► Build • CodeBuild

► Deploy • CodeDeploy

▼ Pipeline • CodePipeline

Getting started

Pipelines

► Settings

Developer Tools > CodePipeline > Pipelines

Pipelines [Info](#)

↺

View history

Re

Q

	Name	Most recent execution	Latest source revision
<input type="radio"/>	Training	✓ Succeeded	Source – ba5085f6:

Pipeline chạy thành công ta truy cập vào IP public của instance truy cập web thành công.

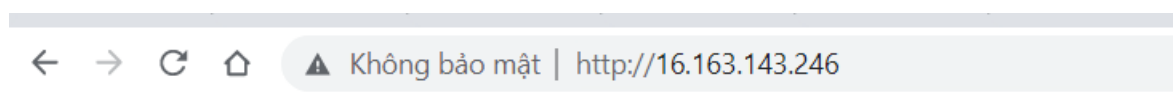
Training DevOps 2022

Chỉ sửa file index.html chờ pipeline chạy và kiểm tra kết quả:

```
Training > <> index.html > h1
1 <h1>Training DevOps 2022</h1>
2 <h1>Demo Codepipeline</h1>
```

44

The screenshot displays the AWS CodePipeline console interface. On the left, a sidebar menu under 'Developer Tools' shows 'CodePipeline' selected. The main area shows a pipeline execution with two stages: 'Build' and 'Deploy', both marked as 'Succeeded'. The 'Build' stage uses 'AWS CodeBuild' and the 'Deploy' stage uses 'AWS CodeDeploy'. Both stages completed 12 minutes ago. The pipeline execution ID is '0f6f6059-109f-402e-bcbd-ba316973bca8'. A 'Disable transition' button is visible above each stage.



Training DevOps 2022

Demo Codepipeline

Pipeline đã tự động chạy khi thay đổi source code.

Hoàn thành bài lab.

Objectives:

- Use CodeCommit, CodeBuild, CodePipeline in CodePipeline

Problem Descriptions:

Assumptions:

Technical Requirements:

- Git, AWS

Questions to answer:

Estimated Time to complete: 700 mins

Day 8, 9. Unit 5: Jenkins

Assignment :

Jenkins Guidelines

1. Install Jenkins

Có nhiều cách để cài đặt Jenkins như sử dụng Docker, Kubernetes, cài đặt trên các hệ điều hành như Linux, macOS, Windows, ... Ở đây mình sẽ cài Jenkins trên máy ảo chạy Ubuntu.

Sau khi tạo instance Ubuntu trên AWS ta ssh vào instance. Vì Jenkins viết bằng java nên để chạy được Jenkins ta phải cài đặt Java trên Ubuntu. Ta sử dụng quyền root.

```
sudo su -  
sudo apt-get update  
sudo apt-get install default-jre -y  
sudo apt-get install default-jdk -y
```

Cài đặt Jenkins:

```
apt-get install wget  
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key |  
  sudo apt-key add -  
sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > \  
/etc/apt/sources.list.d/jenkins.list'  
apt-get update  
apt-get install jenkins -y
```

Khởi động Jenkins:

```
sudo systemctl enable jenkins  
sudo systemctl start jenkins
```

```
root@ip-172-31-8-189:~# sudo systemctl enable jenkins  
jenkins.service is not a native service, redirecting to systemd-sysv-install.  
Executing: /lib/systemd/systemd-sysv-install enable jenkins  
root@ip-172-31-8-189:~# sudo systemctl start jenkins  
root@ip-172-31-8-189:~# █
```

Bây giờ chúng ta sẽ kiểm tra xem Jenkins đã chạy hay chưa bằng cách truy cập: http://IP_public:8080. Ta có thể kiểm tra IP_public của instance bằng cách kiểm tra trên UI của AWS hoặc chạy lệnh:

```
curl ident.me
```

```
root@ip-172-31-8-189:~# curl ident.me  
16.162.106.25root@ip-172-31-8-189:~# █
```

http://16.162.106.25:8080/login?from=%2F

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

Chúng ta sẽ lấy mật khẩu mặc định của Jenkins trong đường dẫn màu đỏ.

```
root@ip-172-31-8-189:~# cat /var/lib/jenkins/secrets/initialAdminPassword
c3d7c1edf294445cba954f6fede30a60
root@ip-172-31-8-189:~#
```

Copy mật khẩu vào Administrator password để tiếp tục cài đặt Jenkins. Chúng ta có thể chọn suggested plugins hoặc cài các plugins theo nhu cầu.

http://16.162.106.25:8080

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Chờ cài đặt hoàn tất:

Getting Started

✓ Folders Plugin	✓ OWASP Markup Formatter Plugin	✓ Build Timeout	✓ Credentials Binding Plugin	<div>***** Jenkins *****</div> <div>** Jenkins Apache HttpComponents Client 4.x API Plugin</div> <div>** Windows Slaves Plugin</div> <div>** Display URL API</div> <div>Jenkins Mailer Plugin</div> <div>Matrix Authorization Strategy Plugin</div> <div>** Jenkins JSch dependency plugin</div> <div>** Jenkins Git client plugin</div> <div>** Jenkins GIT server Plugin</div> <div>** Pipeline: Shared Groovy Libraries</div> <div>** Branch API Plugin</div> <div>** Pipeline: Multibranch</div> <div>** Authentication Tokens API Plugin</div> <div>** Docker Commons Plugin</div> <div>** Docker Pipeline</div> <div>** Pipeline: Stage Tags Metadata</div> <div>** Pipeline: Declarative Agent API</div> <div>** Pipeline: Basic Steps</div> <div>** Pipeline: Declarative Pipeline</div> <div>Pipeline</div> <div>** GitHub API Plugin</div> <div>Jenkins Git plugin</div> <div>** GitHub plugin</div> <div>** - required dependency</div>
✓ Timestamper	✓ Workspace Cleanup Plugin	✓ Ant Plugin	✓ Gradle Plugin	
✓ Pipeline	🔄 GitHub Branch Source Plugin	🔄 Pipeline: GitHub Groovy Libraries	✓ Pipeline: Stage View Plugin	
✓ Git plugin	🔄 Subversion Plug-in	🔄 SSH Slaves plugin	✓ Matrix Authorization Strategy Plugin	
🔄 PAM Authentication plugin	🔄 LDAP Plugin	🔄 Email Extension Plugin	✓ Mailer Plugin	

Jenkins 2.73.3

Tạo tài khoản để truy cập vào Jenkins, chúng ta điền đầy đủ thông tin vào form:

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Cài đặt Jenkins URL:

Instance Configuration

Jenkins URL:

`http://16.162.106.25:8080/`

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Giao diện của Jenkins:

The screenshot shows the Jenkins web interface. At the top is a black header with the Jenkins logo on the left, a search bar in the center, and a notification icon on the right. Below the header is a light gray sidebar on the left containing a 'Dashboard' link and a list of menu items: 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Lockable Resources', and 'New View'. The main content area on the right has a 'Welcome to Jenkins!' heading, followed by a paragraph explaining the page's purpose. Below this are two sections: 'Start building your software project' with a 'Create a job' button, and 'Set up a distributed build' with buttons for 'Set up an agent', 'Configure a cloud', and a link to 'Learn more about distributed builds'. At the bottom left of the main area, there are two expandable sections: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing '1 Idle').


2. Run the first Job in Jenkins

Chúng ta chọn New Item trong Dashboard của Jenkins, điền tên và chọn **Freestyle Project**.


Enter an item name

Test


» Required field

**Freestyle project**


This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**


Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**


Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.

Phần Build chọn Execute Shell và điền một số command:

Build


**Execute shell**

Command


```
echo "hello!!!"  
echo "DevOps 2022"
```


See [the list of available environment variables](#)


Lưu lại cài đặt và bấm Build Now để run Job. Chọn Console Output để xem kết quả:


 [Back to Project](#)

 [Status](#)

 [Changes](#)

 **Console Output**

 [View as plain text](#)

 [Edit Build Information](#)

 [Delete build '#1'](#)

Console Output

Started by user

Running as SYSTEM

Building in workspace /var/lib/jenkins/workspace/Test

[Test] \$ /bin/sh -xe /tmp/jenkins18274520624757078829.sh

+ echo hello!!!

hello!!!

+ echo DevOps 2022

DevOps 2022

Finished: SUCCESS

Thông báo cho thấy chúng ta đã chạy thành công.

3. Manage Plugins

Để quản lý các plugin của Jenkins ta vào: **Manage Jenkins -> Manage Plugins.**

UpdatesAvailableInstalledAdvanced

Install ↑	Name	Version	Released
<input type="checkbox"/>	Javadoc	1.6	1 yr 4 mo ago
<input type="checkbox"/>	Authentication Tokens API This plugin provides an API for converting credentials into authentication tokens in Jenkins.	1.4	1 yr 5 mo ago
<input type="checkbox"/>	Maven Integration Build Tools This plug-in provides, for better and for worse, a deep integration of Jenkins and Maven: Automatic triggers between projects depending on SNAPSHOTS, automated configuration of various Jenkins publishers (JUnit, ...).	3.15.1	1 mo 3 days ago
<input type="checkbox"/>	Docker Commons api-plugindockerLibrary plugins (for use by other plugins) Provides the common shared functionality for various Docker-related plugins.	1.17	1 yr 5 mo ago
<input type="checkbox"/>	JavaScript GUI Lib: jQuery bundles (jQuery and jQuery UI) User Interface JavaScript GUI Lib: jQuery bundles (jQuery and jQuery UI) plugin.	1.2.1	5 yr 9 mo ago

WMI Windows Agents

Install without restart

Download now and install after restart

Update information obtained: 1 hr 10 min ago

Check now

Ở đây ta có 4 bảng:

- Updates: là các plugin đã có phiên bản mới
- Available: là các plugin có trong Jenkins chưa cài đặt
- Installed: là các plugin đã cài đặt
- Advanced: là các plugin bên ngoài Jenkins

Để cài đặt Plugin ta tìm kiếm plugin cần cài đặt, chọn và nhấn Install. Ví dụ như dưới hình ta đang cài đặt plugin về Docker và Docker Pipeline. Ta có thể chọn cài đặt plugin không cần khởi động lại Jenkins hoặc cài đặt và khởi động lại Jenkins. Thường ta sẽ cài đặt xong sẽ khởi động lại để cho plugin hoạt động tốt hơn.

UpdatesAvailableInstalledAdvanced

Install ↑	Name	Version	Released
<input checked="" type="checkbox"/>	Docker Cloud ProvidersCluster Managementdocker This plugin integrates Jenkins with Docker	1.2.3	3 mo 19 days ago
<input type="checkbox"/>	Docker Commons api-plugindockerLibrary plugins (for use by other plugins) Provides the common shared functionality for various Docker-related plugins.	1.17	1 yr 5 mo ago
<input checked="" type="checkbox"/>	Docker Pipeline DeploymentDevOpsdockerpipeline Build and use Docker containers from pipelines.	1.26	9 mo 15 days ago
<input type="checkbox"/>	Docker API api-plugindocker This plugin provides docker-java API for other plugins. <div>This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.</div>	3.1.5.2	1 yr 8 mo ago

Install without restart

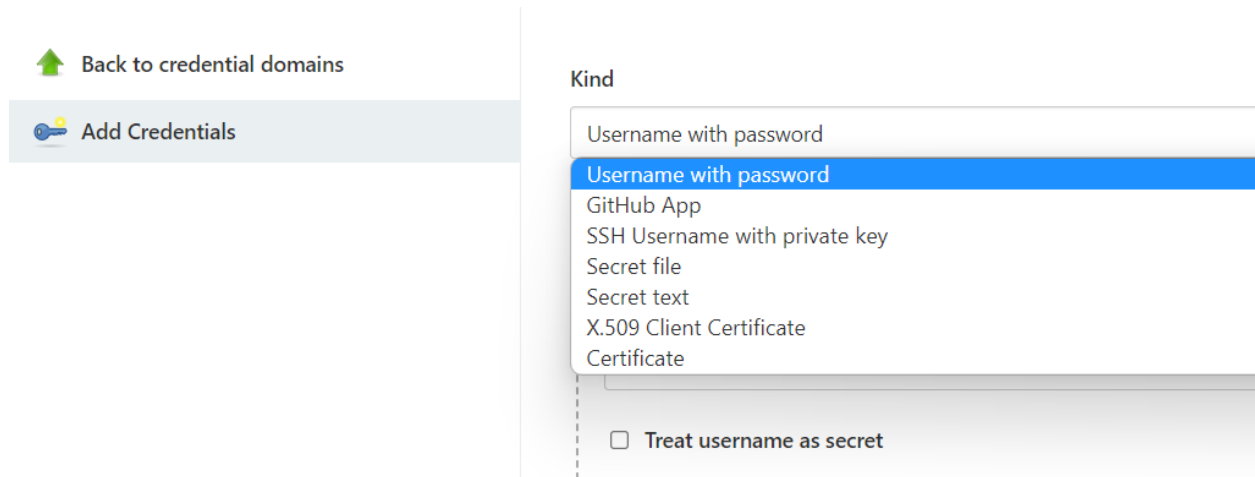
Download now and install after restart

Update information obtained: 1 hr 10 min ago

Check now

4. Manage Credentials

Add Credentials: **Manage Jenkins** -> **Manage Credentials** -> trong Stores scoped to Jenkins chọn **Jenkins** -> **Global Credentials** -> **Add Credentials**.



Back to credential domains

Add Credentials

Kind

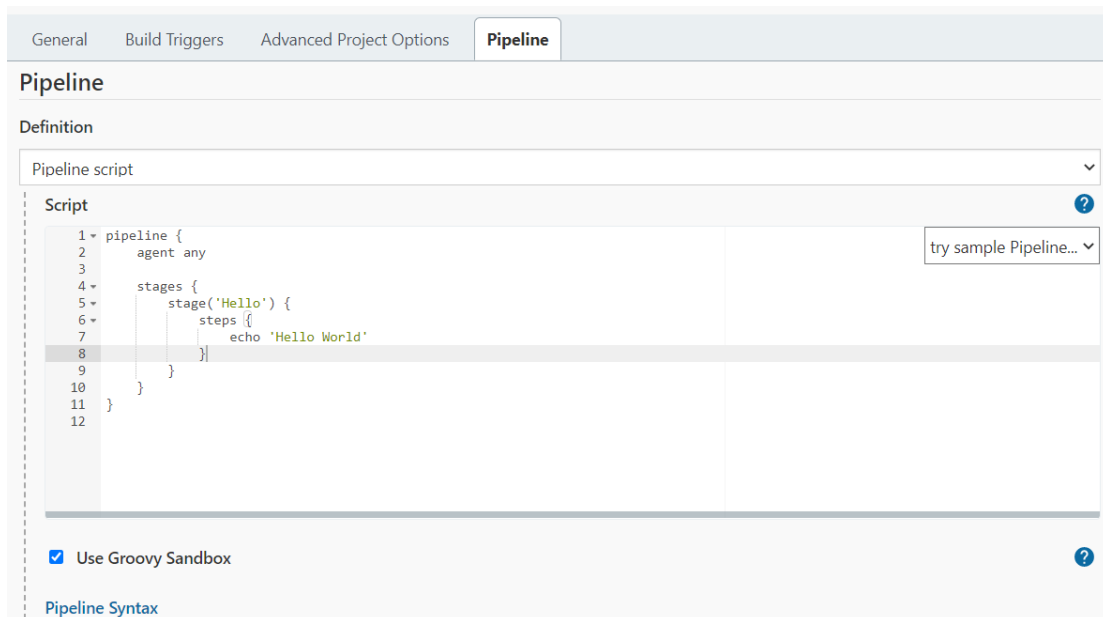
- Username with password
- Username with password
- GitHub App
- SSH Username with private key
- Secret file
- Secret text
- X.509 Client Certificate
- Certificate

☐ Treat username as secret

Có nhiều kiểu dữ liệu trong credentials ta có thể chọn. Điền thông tin theo form và lưu lại.

5. Jenkins pipeline

Jenkinsfile có 2 kiểu là **Declarative** và **Scripted**. Chúng ta sẽ tìm hiểu về kiểu **Declarative**. Để tạo pipeline ta vào Dashboard của Jenkins chọn **New Item** -> pipeline. Phần **Pipeline Definition** chọn **pipeline script** và chọn mẫu script “**Hello World**” để chạy thử.



The screenshot shows the Jenkins Pipeline Definition page. The 'Definition' dropdown is set to 'Pipeline script'. The 'Script' tab is active, displaying a sample pipeline script:

```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Hello') {
6       steps {
7         echo 'Hello World'
8       }
9     }
10  }
11 }
12
```

Below the script, there is a checkbox labeled 'Use Groovy Sandbox' which is checked. A 'try sample Pipeline...' button is also visible. At the bottom, there is a link for 'Pipeline Syntax'.

Các cú pháp trong pipeline ta có thể tìm trong Pipeline Syntax. Ví dụ, ta chạy lệnh echo “just test” thì Pipeline Syntax sẽ tự động Generate ra Pipeline Script .

Steps

Sample Step

sh: Shell Script

sh

Shell Script

echo "just test"

Generate Pipeline Script

sh 'echo "just test"'

Sau khi khai báo xong ta lưu lại và chạy pipeline và kiểm tra kết quả pipeline đã chạy thành công.

```
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/dev
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Hello)
[Pipeline] echo
Hello World
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Tương tự như khi tạo **Pipeline script** nhưng **Pipeline script from SCM** sẽ lấy nguồn từ Git và cần khai báo repository. Các repository private cần thêm credential để access vào. Khai báo đường dẫn của Jenkinfile tại Path và có thể thay đổi tên Jenkinfile theo source code.

Tạo Jenkinsfile với nội dung sau và push lên repository trên github:

```
pipeline {
  agent any
  stages {
    stage('Demo') {
      steps {
        sh 'echo "just test"'
      }
    }
  }
}
```

Tạo một pipeline và cấu hình như dưới hình:

Pipeline

Definition

Pipeline script from SCM

SCM

None

Script Path

Jenkinsfile

☒ Lightweight checkout

[Pipeline Syntax](#)

Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com, /Training.git

Credentials

- none - Add

Branches to build

Branch Specifier (blank for 'any')

*/main

Lưu lại cấu hình và Build để kiểm tra kết quả:

```

Console Output
Started by user
Obtained Jenkinsfile from git https://github.com/Thuong1234/Training.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/Testcode
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Testcode/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/tk/Training.git # timeout=10
Fetching upstream changes from https://github.com/tk /Training.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'

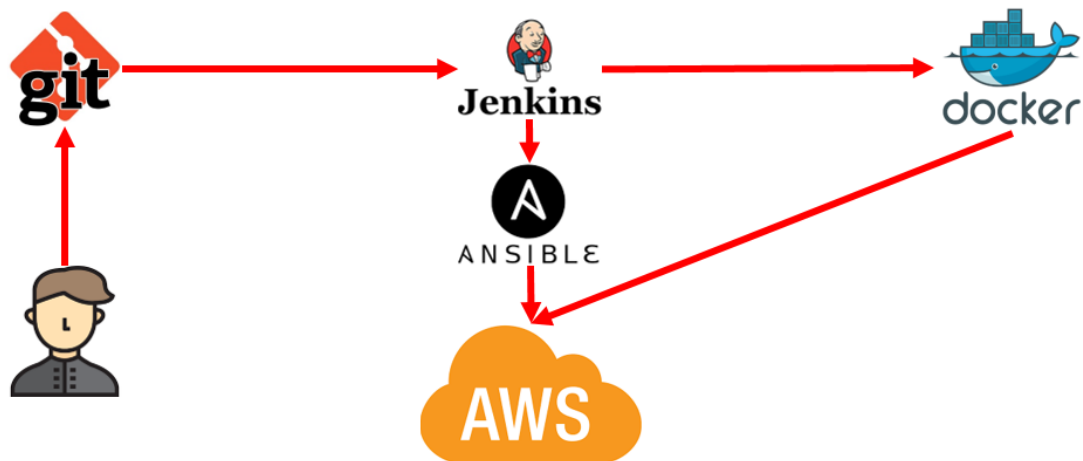
```



```
> git fetch --tags --force --progress -- https://github.com/tk/Training.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 4c12c716eb8015633e7566d3ea61912c22f54810
(refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 4c12c716eb8015633e7566d3ea61912c22f54810 # timeout=10
Commit message: "first commit"
> git rev-list --no-walk b22ee0c3de695e6d5e9df3295e98d72c859c3e80 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Demo)
[Pipeline] sh
+ echo just test
just test
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Kết quả pipeline chạy thành công và đã thực hiện script trên.

6. Run Jenkins pipeline with Ansible, Docker



Mô tả bài Lab: Ta sẽ tạo Jenkins pipeline tự động trigger sự thay đổi trong source từ GitHub và build image từ source sau đó push image lên Docker Hub. Sử dụng ansible để deploy image được tạo lên server trên AWS.

Các bước thực hiện:

- Tạo 1 instance trên AWS (Module AWS)
- Cài đặt Docker, Docker-compose trên máy chạy Jenkins và instance deploy app (Module Linux)
- Tạo repository trên Github (Module Git)
- Tạo repository trên máy tính local (Module Git)
- Viết Jenkinfile và source code và push source code lên Github
- Cài đặt các plugin cần thiết
- Thêm các credentials cần thiết
- Cài đặt GitHub server trên Jenkins (Tự động tạo Webhook)
- Tạo Jenkins Pipeline
- Run Jenkins Pipeline và kiểm tra kết quả
- Thay đổi nội dung source code vào push lại để kiểm sự thay đổi


Các bước cơ bản đã được học ở các học phần khác. Ta bắt đầu bài Lab:

- **Viết Jenkinfile và source code và push source code lên Github**

Nội dung source code gồm các file như sau:

Link: <https://github.com/Thuong1234/Training.git>


✓ Training

 docker-compose.yml

 Dockerfile

 hosts

 index.html

 Jenkinsfile

 playbook.yml

```
//docker-compose.yml
version: "3.3"
services:
  app:
    image: ddthuong/nginx:latest
    ports:
      - "80:80"
```

```
//Dockerfile
FROM nginx:1.13.9-alpine
COPY index.html /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

```
//hosts
[server]
18.167.42.94 ansible_user="ubuntu" ansible_python_interpreter='/usr/bin/env python3'
ansible_ssh_extra_args='-o StrictHostKeyChecking=no'
```

```
//index.html
<h1>DevOps Training 2022</h1>
```

```
//Jenkinsfile
pipeline {
  agent any
  environment{
    DOCKER_IMAGE = "ddthuong/nginx"
  }
}
```

```

stages {
  stage("Build"){
    options {
      timeout(time: 10, unit: 'MINUTES')
    }
    environment {
      DOCKER_TAG="${GIT_BRANCH.tokenize('/').pop()}-${GIT_COMMIT.substring(0,7)}"
    }
    steps {
      sh """
        docker build -t ${DOCKER_IMAGE}:${DOCKER_TAG} .
        docker tag ${DOCKER_IMAGE}:${DOCKER_TAG}
        ${DOCKER_IMAGE}:latest
        docker image ls | grep ${DOCKER_IMAGE}"
      withCredentials([usernamePassword(credentialsId: 'docker-hub',
usernameVariable: 'DOCKER_USERNAME', passwordVariable:
'DOCKER_PASSWORD')]) {
        sh 'echo $DOCKER_PASSWORD | docker login --username
$DOCKER_USERNAME --password-stdin'
        sh "docker push ${DOCKER_IMAGE}:${DOCKER_TAG}"
        sh "docker push ${DOCKER_IMAGE}:latest"
      }

      //clean to save disk
      sh "docker image rm ${DOCKER_IMAGE}:${DOCKER_TAG}"
      sh "docker image rm ${DOCKER_IMAGE}:latest"
    }
  }
  stage("Deploy"){
    options {
      timeout(time: 10, unit: 'MINUTES')
    }
    steps {
      withCredentials([usernamePassword(credentialsId: 'docker-hub',
usernameVariable: 'DOCKER_USERNAME', passwordVariable:
'DOCKER_PASSWORD')]) {
        ansiblePlaybook(
          credentialsId: 'private_key',
          playbook: 'playbook.yml',
          inventory: 'hosts',
          become: 'yes',
          extraVars: [
            DOCKER_USERNAME: "${DOCKER_USERNAME}",
            DOCKER_PASSWORD: "${DOCKER_PASSWORD}"
          ]
        )
      }
    }
  }
}

```

```

    }
  }
  post {
    success {
      echo "SUCCESSFULL"
    }
    failure {
      echo "FAILED"
    }
  }
}

```

```

//playbook.yml
---
- hosts: server
  remote_user: ubuntu
  become: yes
  tasks:
    - name: Install Docker
      apt:
        name:
          - docker
          - docker-compose
        state: present
    - name: Login DockerHub
      shell: docker login -u "{{DOCKER_USERNAME}}" -p "{{DOCKER_PASSWORD}}"
    - name: Copy file
      copy:
        src: docker-compose.yaml
        dest: /home/ubuntu/
    - name: Restart docker-compose
      shell: |
        docker-compose stop
        docker-compose rm -f
        docker-compose pull
        docker-compose up -d
  ...

```

▪ Cài đặt các plugin cần thiết

Trong bài lab này chúng ta sử dụng Jenkins kết hợp ansible chạy pipeline deploy app chạy trên docker nên ta sẽ cài đặt các plugin: **Docker plugin, Docker Pipeline, Ansible plugin, GitHub Intergration Plugin.**



Docker Pipeline

Build and use Docker containers from pipelines.



Docker plugin

This plugin integrates Jenkins with [Docker](#)



Ansible plugin

Invoke [Ansible](#) Ad-Hoc commands and playbooks.



GitHub Integration Plugin

GitHub Integration Plugin for Jenkins

- **Thêm các credentials cần thiết**

Trong bài Lab này chúng ta cần credential cho Docker Hub để lưu trữ image và credential private key để access vào instance deploy app.

Credentials của Docker Hub dạng **Username with password**:

Scope


Global (Jenkins, nodes, items, all child items, etc)

Username

ddthuong

☐ Treat username as secret

Password

 Concealed

ID

docker-hub

Credential private key dạng **SSH Username with private key**:

Scope

Global (Jenkins, nodes, items, all child items, etc)

ID

private_key

Description

Username


ubuntu

☐ Treat username as secret

Private Key

☒ Enter directly

Key

 Concealed for Confidentiality

- **Cài đặt GitHub server trên Jenkins (Tự động tạo Webhook)**

Để cài đặt GitHub server ta cần tạo Personal access tokens trên GitHub chúng ta đã push source lên. Ta vào GitHub tại mục **Setting** tài khoản -> **Developer setting** -> **Personal access token** -> **Generate new token** và chọn các tùy chọn như ảnh dưới:

<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input checked="" type="checkbox"/> write:repo_hook	Write repository hooks
<input checked="" type="checkbox"/> read:repo_hook	Read repository hooks
<input checked="" type="checkbox"/> admin:org_hook	Full control of organization hooks
<input type="checkbox"/> gist	Create gists
<input checked="" type="checkbox"/> notifications	Access notifications
<input type="checkbox"/> user	Update ALL user data
<input type="checkbox"/> read:user	Read ALL user profile data
<input type="checkbox"/> user:email	Access user email addresses (read-only)
<input type="checkbox"/> user:follow	Follow and unfollow users
<input type="checkbox"/> delete_repo	Delete repositories
<input checked="" type="checkbox"/> write:discussion	Read and write team discussions
<input checked="" type="checkbox"/> read:discussion	Read team discussions

Tiếp theo ta vào **Dashboard** Jenkins chọn **Manage Jenkins -> Configure System** tìm mục GitHub Servers và điền các thông tin, tại mục Credentials ta thêm vào tài khoản GitHub với password là token vừa tạo ở trên sau đó bấm **Test connection** để kiểm tra kết nối.

GitHub

GitHub Servers

GitHub Server

Name

Thuong1234

API URL

https://api.github.com

Credentials

github

Add

Credentials verified for user Thuong1234, rate limit: 4998

Test connection

Manage hooks

Advanced...

Delete

Add GitHub Server

- **Tao Jenkins Pipeline**

Chúng ta điền các thông tin như hình dưới.

Enter an item name

» Required field



Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK

General

[Build Triggers](#)[Advanced Project Options](#)[Pipeline](#)

Description

[Plain text] [Preview](#)

- ☐ Discard old builds
- ☐ Do not allow concurrent builds
- ☐ Do not allow the pipeline to resume if the controller restarts

☒ GitHub project

Project url

Advanced...

- ☐ Pipeline speed/durability override
- ☐ Preserve stashes from completed builds
- ☐ This project is parameterized
- ☐ Throttle builds

Save

Apply

General **Build Triggers** Advanced Project Options Pipeline

Build Triggers

- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ Generic Webhook Trigger
- ☐ GitHub Branches
- ☐ GitHub Pull Requests
- ☒ GitHub hook trigger for GITScm polling
- ☐ Poll SCM
- ☐ Disable this project
- ☐ Quiet period
- ☐ Trigger builds remotely (e.g., from scripts)

Ở đây nếu repository trên GitHub của bạn là private thì cần thêm Credentials để xác thực.

General Build Triggers Advanced Project Options **Pipeline**

Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/Thuong1234/Training.git

Credentials

Thuong1234/***** Add

Cấu hình Branch chính trong repository và định nghĩa Jenkinsfile theo tên file. Sau đó lưu lại và build.

General Build Triggers Advanced Project Options **Pipeline**

Branches to build

Branch Specifier (blank for 'any')

*/main

Add Branch

Repository browser

(Auto)

Additional Behaviours

Add

Script Path

Jenkinsfile

☒ Lightweight checkout

Changes
 Build Now
 Configure
 Delete Pipeline
 Full Stage View
 GitHub
 Rename
 Pipeline Syntax
 GitHub Hook Log

Recent Changes

Stage View

Average stage times:
(Average full run time: ~22s)

#1

Dec 09 10:48

No Changes

Declarative: Checkout SCM	Build	Deploy	Declarative: Post Actions
581ms	12s	7s	52ms

Permalinks

Build History

trend ^

#1

Dec 9, 2021, 3:48 AM

[Atom feed for all](#)
[Atom feed for failures](#)

Pipeline chạy thành công chúng ta thử truy cập vào địa chỉ public IP của instance deploy app kiểm tra kết quả đã chạy thành công.

← → ↺ 🏠 ⚠ Không bảo mật | http://18.167.42.94

DevOps Training 2022

Test webhook

Kiểm tra webhook đã tự động được tạo trên GitHub.

Options
 Manage access
 Security & analysis
 Branches
 Webhooks
 Notifications
 Integrations
 Deploy keys

Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

• http://16.162.90.207:8080/github... (push)

Edit Delete

Ta thử thay đổi nội dung file index.html và push lại code lên repository. Kiểm tra trong giao diện **Build History** Jenkins nhận được sự thay đổi trong source code của chúng ta và đang tự động build.

Delete Pipeline

Full Stage View

GitHub

Rename

Pipeline Syntax

GitHub Hook Log

Build History

trend ^

find

#3

(pending—In the quiet period. Expires in 4.7 sec)

#2

Dec 9, 2021, 3:53 AM

#1

Dec 9, 2021, 3:48 AM

Configure

Delete Pipeline

Full Stage View

GitHub

Rename

Pipeline Syntax

GitHub Hook Log

Build History

trend ^

find

#3

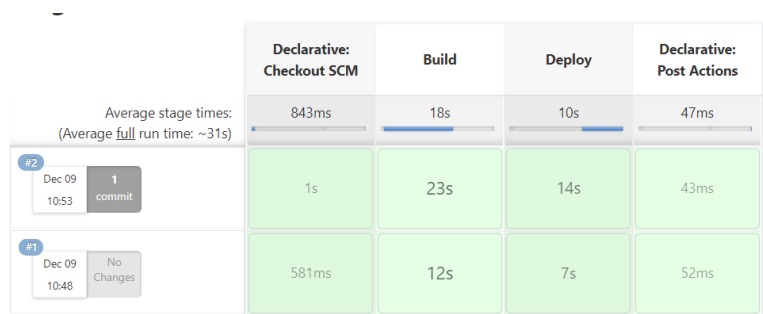
(pending—Finished waiting)

#2

Dec 9, 2021, 3:53 AM

#1

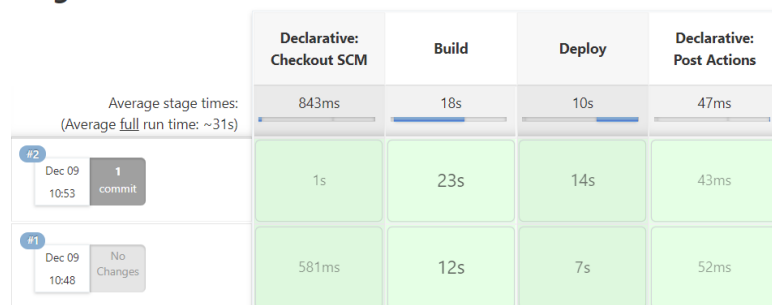
Dec 9, 2021, 3:48 AM



Permalinks

- Last build (#2), 3 min 49 sec ago
- Last stable build (#2), 3 min 49 sec ago
- Last successful build (#2), 3 min 49 sec ago
- Last completed build (#2), 3 min 49 sec ago

Stage View



Permalinks

- Last build (#2), 3 min 49 sec ago
- Last stable build (#2), 3 min 49 sec ago
- Last successful build (#2), 3 min 49 sec ago
- Last completed build (#2), 3 min 49 sec ago

Chờ cho pipeline chạy xong và kiểm tra kết quả nội dung index đã được thay đổi.

← → ↺ ⌂ ⚠ Không bảo mật | http://18.167.42.94

DevOps Training 2022

Test webhook

Change source code

Ta tiếp tục thử lại lần nữa xem pipeline chạy ổn định hay chưa.

Delete Pipeline
Full Stage View
GitHub
Rename
Pipeline Syntax
GitHub Hook Log

Build History
trend
find
#3 Dec 9, 2021, 3:57 AM
#2 Dec 9, 2021, 3:53 AM
#1 Dec 9, 2021, 3:48 AM

Delete Pipeline
Full Stage View
GitHub
Rename
Pipeline Syntax
GitHub Hook Log

Build History
trend
find
#3 Dec 9, 2021, 3:57 AM
#2 Dec 9, 2021, 3:53 AM
#1 Dec 9, 2021, 3:48 AM

Average stage times:
(Average full run time: ~31s)

	Declarative: Checkout SCM	Build	Deploy	Declarative: Post Actions
#3 Dec 09 10:57 1 commit	912ms			
#2 Dec 09 10:53 1 commit	1s	23s	14s	43ms
#1 Dec 09 10:48 No Changes	581ms	12s	7s	52ms

Permalinks

Average stage times:
(Average full run time: ~31s)

	Declarative: Checkout SCM	Build	Deploy	Declarative: Post Actions
#3 Dec 09 10:57 1 commit	912ms	22s	7s	42ms
#2 Dec 09 10:53 1 commit	1s	23s	14s	43ms
#1 Dec 09 10:48 No Changes	581ms	12s	7s	52ms

← → ↺ 🏠 ⚠ Không bảo mật | http://18.167.42.94

DevOps Training 2022

Test webhook

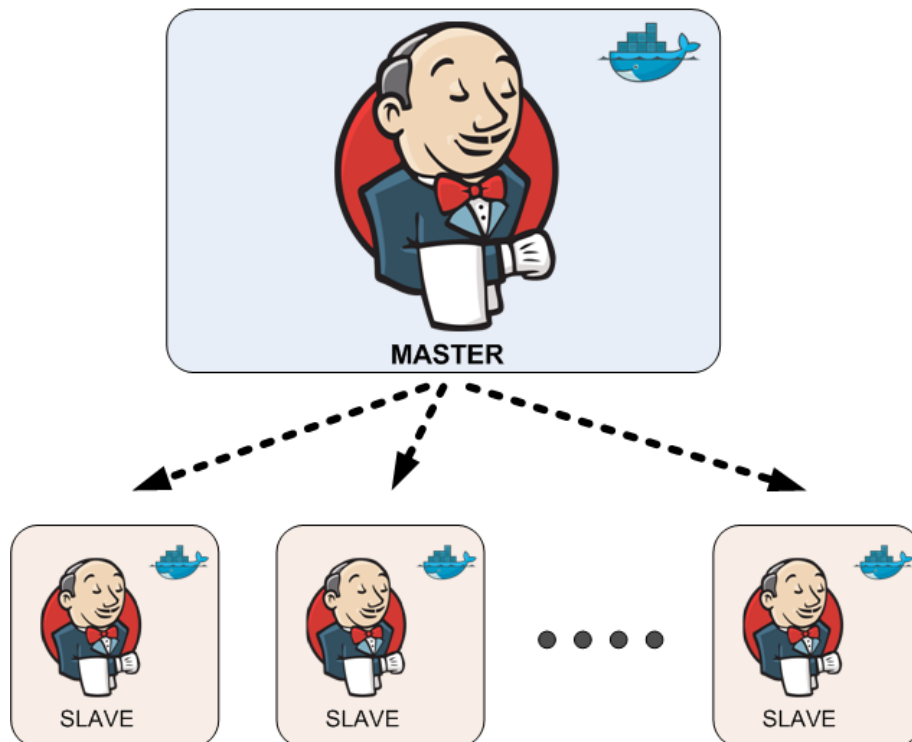
Change source code 1

Kết quả: pipeline đã tự động build khi chúng ta thay đổi source code.

Hoàn thành bài lab.

7. Jenkins Master Slave

Bài Lab này chúng ta sẽ cấu hình 1 Jenkins Master và 1 Jenkins Slave



Jenkins Master ta sẽ sử dụng máy chạy Jenkins ở bài Lab trước, Jenkins Slave sẽ chạy trên 1 instance khác ta sẽ tạo.

- Tạo 1 instance trên AWS (Module AWS)

Cài đặt java trên Slave:

```
sudo su -  
sudo apt-get update  
sudo apt-get install default-jre -y  
sudo apt-get install default-jdk -y
```

- Thêm Slave vào known_hosts on Master

Vì public IP bị thay đổi mỗi khi tắt máy nên ở đây ta dùng private IP vì 2 máy cùng trong VPC. Chạy lệnh sau trên máy Master:

```
$ ssh-keyscan -H IP_Slave >> /var/lib/jenkins/.ssh/known_hosts  
root@ip-172-31-29-124:~# ssh-keyscan -H 172.31.1.136 >> /var/lib/jenkins/.ssh/known_hosts  
# 172.31.1.136:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.3  
# 172.31.1.136:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.3  
# 172.31.1.136:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.3  
# 172.31.1.136:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.3  
# 172.31.1.136:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.3  
root@ip-172-31-29-124:~#
```

- Cấu hình thêm Slave Node trên Master

Ta vào **Dashboard Jenkins -> Manage Jenkins -> Manage Nodes and Clouds -> New Node** và làm theo hình dưới:

Dashboard
Nodes

Back to Dashboard

Manage Jenkins

New Node

Configure Clouds

Node Monitoring

Build Queue (1)

Test

Build Executor Status

1 Idle

2 Idle

Node name

Slave

Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins do type if no other agent types apply — for example such as when you are adding a physical

OK

Các thông số như Remote root directory là thư mục Jenkins Slave chạy trên máy Slave, Label để phân loại các Slave dùng khi cấu hình tùy chọn các slave được chạy.

Jenkins

Dashboard
Nodes

Back to Dashboard

Manage Jenkins

New Node

Configure Clouds

Node Monitoring

Build Queue (1)

Test

Build Executor Status

1 Idle

2 Idle

Name

Slave

Description

Test

Number of executors

1

Remote root directory

/home/ubuntu/

Labels

Test

Usage

Use this node as much as possible

Launch method

Launch agents via SSH

Ở đây ta thêm địa chỉ IP của host và thêm credentials để access vào host.

Launch method

Launch agents via SSH

Host

172.31.1.136

Credentials

ubuntu

Add

Host Key Verification Strategy

Known hosts file Verification Strategy

Availability

Keep this agent online as much as possible

Node Properties

☐ Disable deferred wipeout on this node

☐ Environment variables

☐ Tool Locations

Save

Lưu lại cấu hình và kiểm tra Slave đã chạy.

```
[12/09/21 08:25:49] [SSH] Checking java version of java
[12/09/21 08:25:49] [SSH] java -version returned 11.0.11.
[12/09/21 08:25:49] [SSH] Starting sftp client.
[12/09/21 08:25:49] [SSH] Copying latest remoting.jar...
[12/09/21 08:25:49] [SSH] Copied 1,507,813 bytes.
Expanded the channel window size to 4MB
[12/09/21 08:25:49] [SSH] Starting agent process: cd "/home/ubuntu" && java -jar remoting.jar -workDir /home/ubuntu -jar-cac
Dec 09, 2021 8:25:49 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/remoting as a remoting work directory
Dec 09, 2021 8:25:49 AM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/ubuntu/remoting
<===[JENKINS REMOTING CAPACITY]===>channel started
Remoting version: 4.10.1
This is a Unix agent
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by jenkins.slaves.StandardOutputSwapper$ChannelSwapper to constructor java.io.FileDes
WARNING: Please consider reporting this to the maintainers of jenkins.slaves.StandardOutputSwapper$ChannelSwapper
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Evacuated stdout
Agent successfully connected and online
```

▪ Cấu hình Job cho Slave chạy và kiểm tra kết quả

Tương tự như tạo Job ở trên trong phần **Restrict where this project can be run** ta điền tên của Slave hoặc Label của Slave Jenkins sẽ tìm và trả về các node hợp lệ.

General

Source Code Management

Build Triggers

Build Environment

Build

Post-build Actions

[Plain text] [Preview](#)

☐ Commit agent's Docker container

☐ Define a Docker template

☐ Discard old builds

☐ GitHub project

☐ This build requires lockable resources

☐ This project is parameterized

☐ Throttle builds

☐ Disable this project

☐ Execute concurrent builds if necessary

☒ Restrict where this project can be run

Label Expression

Test

[Label Test](#) matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.

Advanced...

Thêm **Execute shell** để kiểm tra xem Slave có chạy không.

Build

Execute shell

Command

echo "Slave running!"

See [the list of available environment variables](#)

Build job và kiểm tra thấy Slave đã chạy.

Back to Project
 Status
 Changes
 Console Output
 View as plain text
 Edit Build Information
 Delete build '#1'

Console Output

Started by user **admin**
 Running as SYSTEM
 Building remotely on **Slave** (Test) in workspace /home/ubuntu/workspace/Slave
 [Slave] \$ /bin/sh -xe /tmp/jenkins15162647491852816913.sh
 + echo Slave running!
 Slave running!
 Finished: SUCCESS

Hoàn thành bài Lab.

Objectives:

- Create Jenkins Pipeline

Problem Descriptions:

Assumptions:

Technical Requirements:

Questions to answer:

Estimated Time to complete: 700 mins

Day 8, 9. Unit 6: Prometheus+Grafana +AlertManager

Assignment :

Prometheus + Grafana

1. Prometheus

Cài đặt Prometheus tham khảo tại:

<https://linuxhint.com/install-prometheus-on-ubuntu/>

2. Grafana

Cài đặt Grafana tham khảo tại:

<https://linuxways.net/ubuntu/how-to-install-grafana-on-ubuntu-20-04/>

3. Alert Manager

<https://linuxhint.com/install-configure-prometheus-alert-manager-ubuntu/>

Phần cấu hình rule.yml ta tùy biến như sau để thông báo khi instance down hoặc CPU chạy quá 75%

- name: Instances

```

rules:
- alert: InstanceDown
  expr: up == 0
  for: 10s
  labels:
    severity: page
  # Prometheus templates apply here in the annotation and label fields of the alert.
  annotations:
    description: '{{ $labels.instance }} of job {{ $labels.job }} has been down for more than
10 s.'
    summary: 'Instance {{ $labels.instance }} down'
- alert: CPUUsage
  expr: (100 - (avg by (instance) (irate(node_cpu_seconds_total{job="Server
Root",mode="idle"}[1m])) * 100)) > 75
  for: 1m
  labels:
    severity: page
  annotations:
    summary: '{{ $labels.instance }}: High CPU usage detected'
    description: '{{ $labels.instance }}: CPU usage is above 75% (current value is:
{{ $value }})'

```

Objectives:

- Set up Prometheus, Grafana, AlertManager

Problem Descriptions:

Assumptions:

Technical Requirements:

Questions to answer:

Estimated Time to complete: 480 mins

-- THE END --