

CÁC bước CI/CD,jenkins; docker; IIS,MinIO,Grafana,Traefik

Sơ đồ quy trình CI/CD

[GitHub: Push code] → [Jenkins: CI - Restore, Build, Test] → [IIS: CD - Deploy]

**** Lưu ý : đây là cách làm chỉ áp dụng cho dự án bằng .netCore theo mô hình MVC**

Quá trình CI

B1 : Tạo repository:

- Truy cập github.com, tạo 1 repo mới đặt tên là demo
- Rồi git clone https://github.com/zyond26/Restaurant_cicd.git về

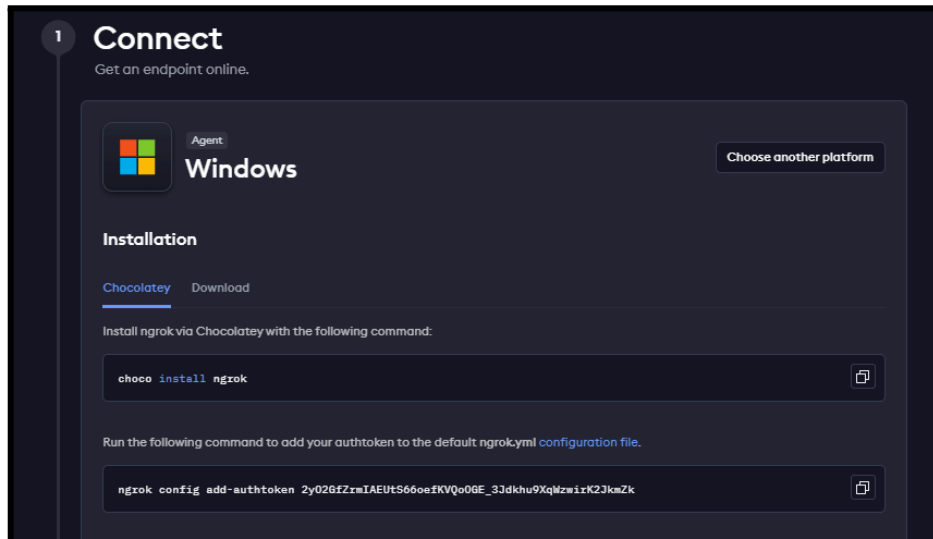
B2: Mở vscode mở thư mục vừa git clone và sau đó mở terminal rồi đẩy code theo lệnh

```
git init
git add .
git commit -m "Initial commit"
git remote add origin https://github.com/<tên github>/demo.git
git branch -M master
git push -u origin master
```

B3: Ánh xạ địa chỉ jenkins ở webhook trong github

- Đã tải ngrok rồi nhé

+) mở ngrok.exe đã tải về xong mở trang chủ ngrok tạo tài khoản xong lấy mã key



Lấy dòng vd :

“ **ngrok config add-authtoken**

2yO2GfZrmIAEUtS66oefKVQoOGE_3Jdkhu9XqWzwirK2wirK2JkmZk”

+) lấy dc user thì thêm vào cmd đg mở của ngrok r ấn enter

+) sau đó bấm " **ngrok http <cổng jenkins>**"

-vd:8099

-> enter rồi hiện ra đoạn link

Thì copy đến chữ app thôi

- vidu: <https://3452-222-252-127-126.ngrok-free.app>

- Mở repo đã tạo ở github ra

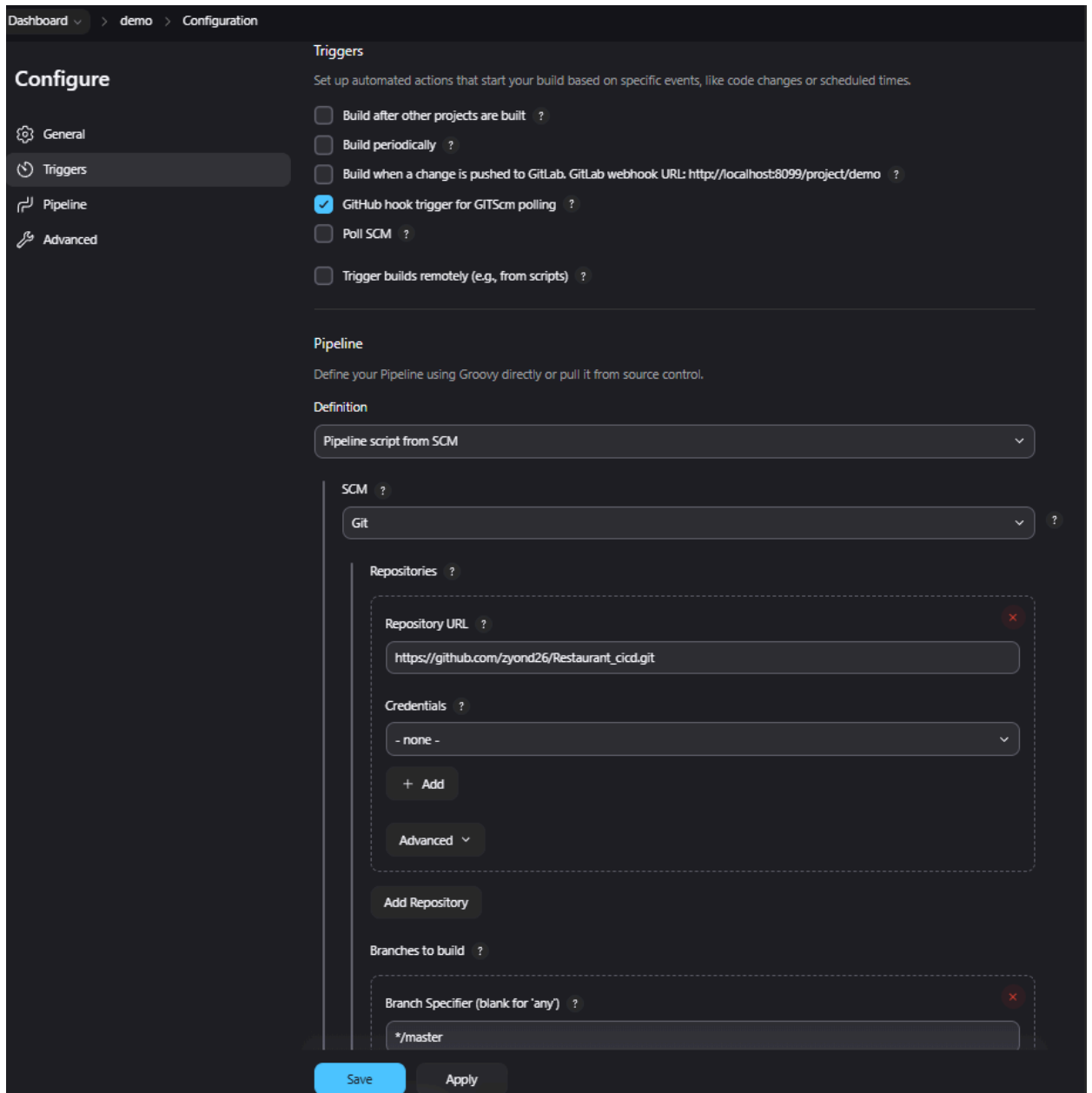
- vào setting của repo đó bấm webhook rồi bấm addwebhook
gắn link: **(link ở ngrok)/github-webhook/**

vidu: <https://3452-222-252-127-126.ngrok-free.app/github-webhook/>

-> bấm save rồi f5 nếu tick xanh là thành công còn ko thì fail

B4 : vào jenkins đăng nhập xong tạo new item

Cấu hình như này



- **New Item** > Tên: WebRestaurant, chọn **Pipeline**.
- **Build Triggers**: Check **GitHub hook trigger for GITScm polling**.
- **Pipeline**: Chọn **Pipeline script from SCM**.
 - SCM: Git.
 - Repository URL: https://github.com/zyond26/Restaurant_cicd.git (ví dụ thôi nha)
 - Branch: master.(hoặc main)
 - Script Path: Jenkinsfile.

+)sau đó ấn save rồi Build Now có tick xanh là được nha

- **Nhớ là tạo 1 file Jenkinsfile xong mới push lên nhé:**
- **Nội dung jenkinsfile là :**

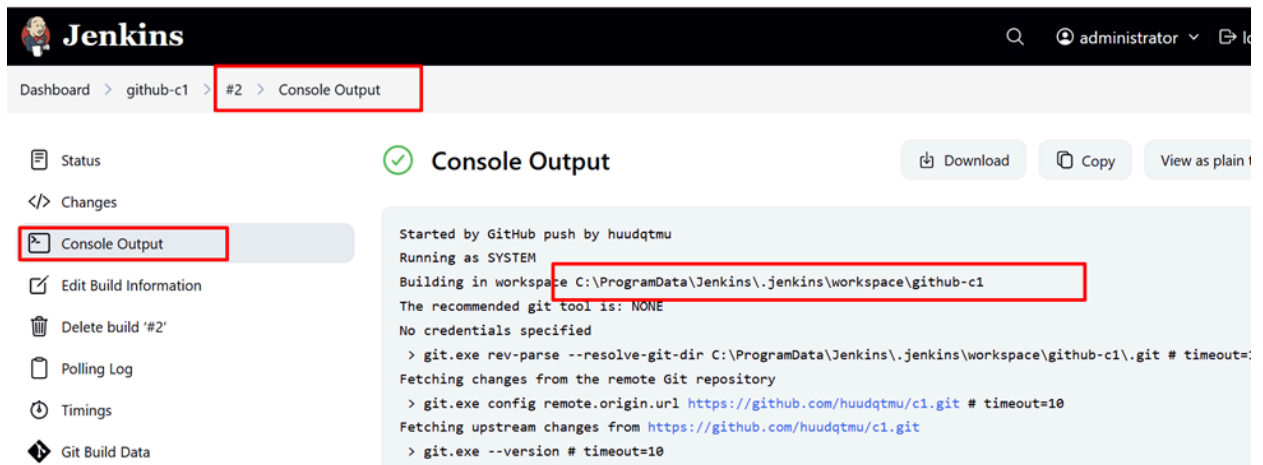
```

pipeline {
  agent any
  stages {
    stage('Clone') {
      steps {
        echo 'Cloning source code'
        git branch: 'main', url: 'đường dẫn đến repo bạn đã
        tạo bên trên trong github'
      }
    }
    stage('Restore Packages') {
      steps {
        echo 'Restoring NuGet packages...'
        bat 'dotnet restore'
      }
    }
    stage('Build') {
      steps {
        echo 'Building the project...'
        bat 'dotnet build --configuration Release'
      }
    }
    stage('Run Tests') {
      steps {
        echo 'Running unit tests...'
        bat 'dotnet test --no-build --verbosity normal'
      }
    }
    stage('Publish to Folder') {
      steps {
        echo 'Cleaning old publish folder...'
        bat 'if exist "%WORKSPACE%\publish" rd /s /q
        "%WORKSPACE%\publish"

        echo 'Publishing to temporary folder...'
        bat 'dotnet publish -c Release -o "%WORKSPACE%\publish"'
      }
    }
  }
}

```

sau đó vào vscode cập nhật gì đó rồi push lên github xem có tự động chạy ở jenkins không; quá trình đẩy mà màu xanh thì thành công Tiếp đó vào console output : tìm link



Jenkins

Dashboard > github-c1 > #2 > Console Output

Status

Changes

Console Output

Edit Build Information

Delete build '#2'

Polling Log

Timings

Git Build Data

Console Output

Download Copy View as plain text

Started by GitHub push by huudqtmu
Running as SYSTEM
Building in workspace C:\ProgramData\Jenkins\.jenkins\workspace\github-c1
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\github-c1\.git # timeout=10
Fetching changes from the remote Git repository
> git.exe config remote.origin.url https://github.com/huudqtmu/c1.git # timeout=10
Fetching upstream changes from https://github.com/huudqtmu/c1.git
> git.exe --version # timeout=10

Copy link khoanh đó rồi paste vào file explore để tìm ra sau đó vào vscode chỉnh thử mấy cái file.txt đẩy lại lên git vào lại file C:/ như kia nếu đổi rồi thìg

=> xong quá trình CI

Quá trình CD

Sau khi CI xong muốn CD thì mở file Jenkinsfile trong dự án ra và thêm:

```
//----- automated deployment to IIS -----  
  
// Copy to IIS Folder  
stage('Copy to IIS Folder') {  
    steps {  
        echo 'Stopping IIS...'  
        bat 'iisreset /stop'  
  
        echo 'Cleaning existing deploy folder...'  
        bat 'if exist C:\\WOR_cicd rd /s /q C:\\WOR_cicd'  
  
        echo 'Creating IIS folder...'  
        bat 'mkdir C:\\WOR_cicd'  
  
        echo 'Copying to IIS folder...'  
        bat 'xcopy /E /Y /I /R "%WORKSPACE%\\publish\\*" "C:\\WOR_cicd\\"  
  
        echo 'Starting IIS again...'  
        bat 'iisreset /start'    }  
}
```

```

    }
  }
  // Ensure IIS Site Exists
  stage('Ensure IIS Site Exists') {
    steps {
      powershell '''
        Import-Module WebAdministration

        $siteName = "WOR_cicd"
        $sitePath = "C:\\\\WOR_cicd"
        $sitePort = 8089

        if (-not (Test-Path "IIS:\\\\Sites\\$siteName")) {
          New-Website -Name $siteName -Port $sitePort -PhysicalPath $sitePath
-Force
        } else {
          Write-Host "Website $siteName already exists"
        }
      '''
    }
  }
}

```

- Push lại lên github kiểm tra jenkins chạy thành công là oke
- rồi mở IIS phần website refresh lại sẽ có web t tên như mình đã đặt
- Bấm vào https local gì đó chạy lên là được
- Sau đó vào code vscode của mình cập nhật gì đó ví dụ như đổi tên web rồi push lại lên github check jenkins thành công xong reload lại web nếu thay đổi là được

⇒ Kết thúc quá trình CD

Quá trình Docker

***** Chạy tự động *****

Bước 1 : Mở Jenkinsfile ra và thêm các lệnh sau :

```

    Nhớ thêm biến môi trường
environment {
    // docker environment variables
    LANG = 'en_US.UTF-8'
    LC_ALL = 'en_US.UTF-8'
    DOCKERHUB_CREDENTIALS =
'a8043e21-320b-4f12-b72e-612d7a93c553' // ID credentials
    IMAGE_NAME = 'zyond/cicd' // name of image on Docker Hub --
create repo on hub.docker
    DOCKER_IMAGE_NAME = 'zyond/cicd' // Docker image name
    DOCKER_TAG = 'latest' // Tag cho Docker image

//----- automated deployment to Docker Hub -----

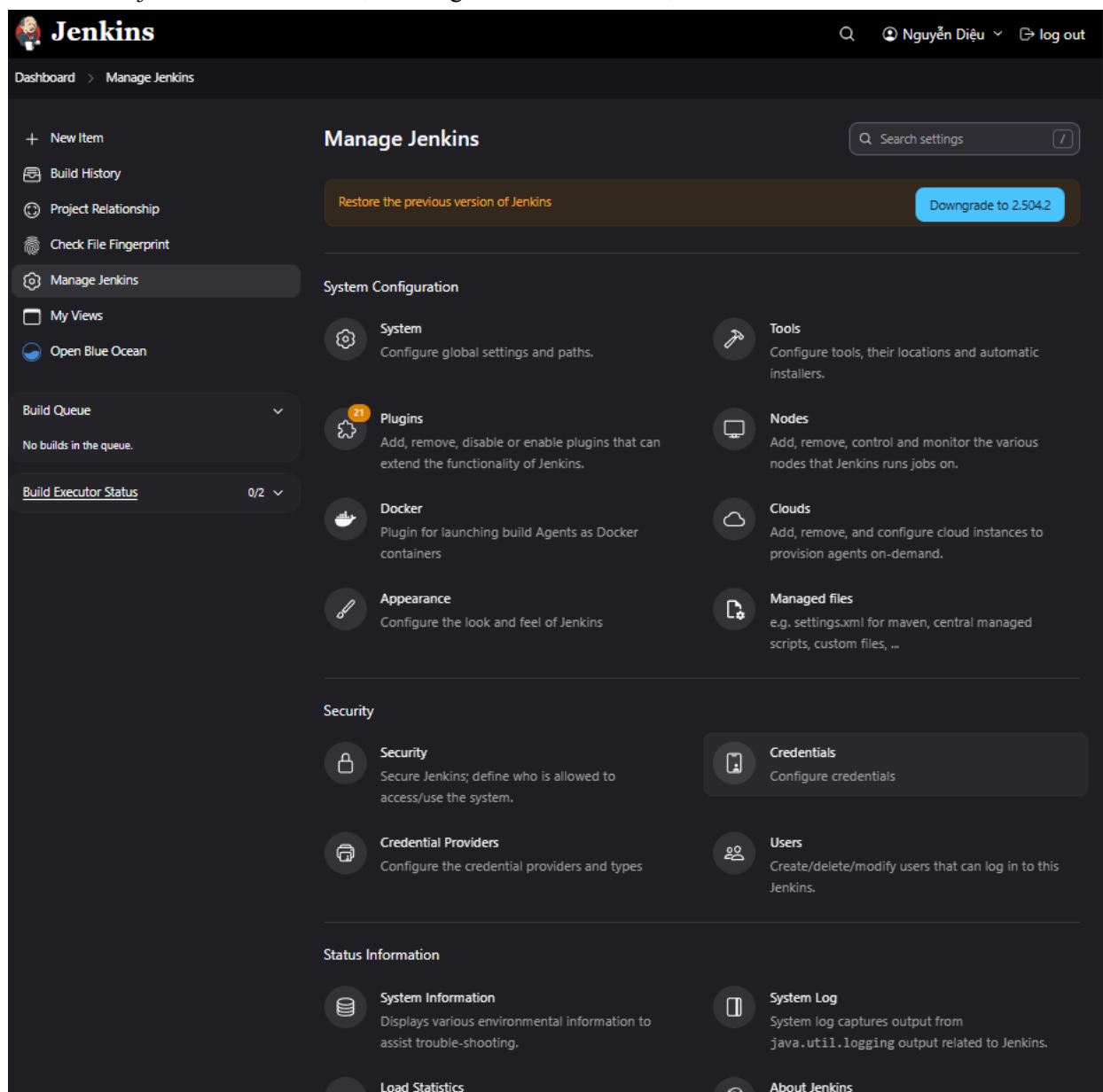
    // Build Docker Image
    stage('Build Docker Image') {
        steps {
            script {
                docker.build("${DOCKER_IMAGE_NAME}:latest")
            }
        }
    }

    // Login to Docker Hub
    stage('Login to Docker Hub') {
        steps {
            script {
                docker.withRegistry('https://index.docker.io/v1/',
DOCKERHUB_CREDENTIALS) {
                    // login Docker Hub credentials
                }
            }
        }
    }

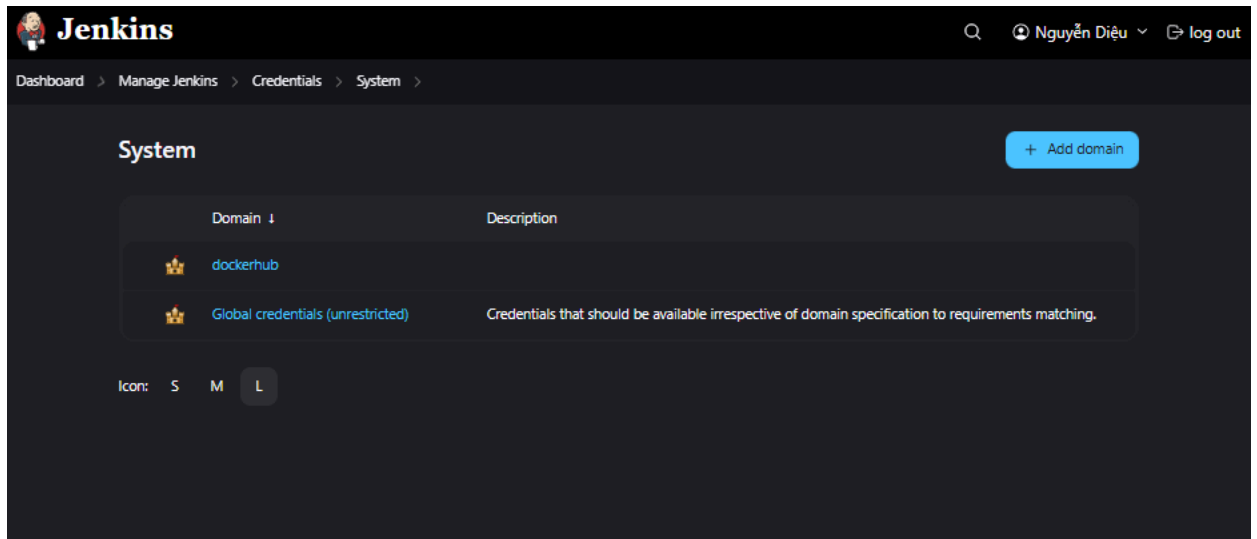
    // Push Docker Image to Docker Hub
    stage('Push Docker Image') {
        steps {
            script {
                docker.withRegistry('https://index.docker.io/v1/',
DOCKERHUB_CREDENTIALS) {
                    docker.image("${DOCKER_IMAGE_NAME}:${DOCKER_TAG}").push()
                }
            }
        }
    }
}

```

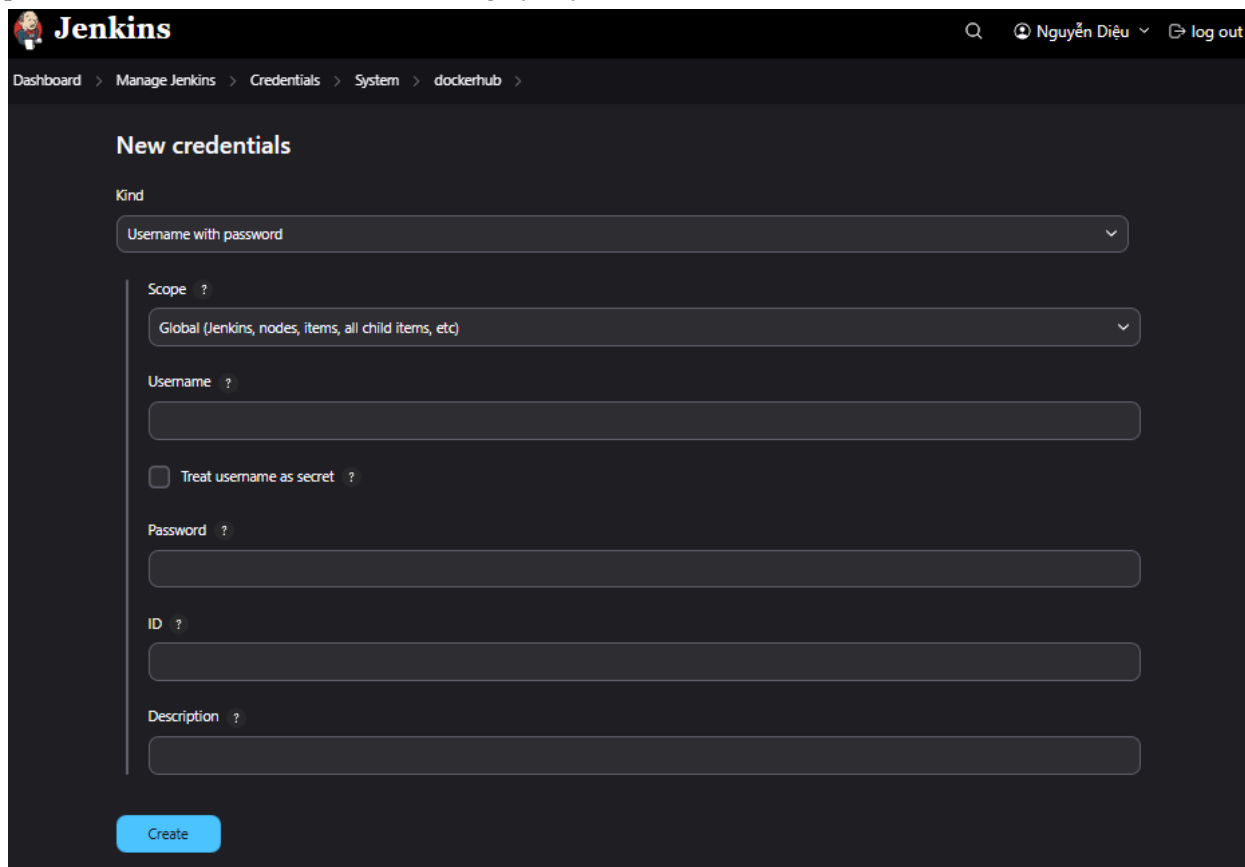
Bước 2: vào jenkins dashboard chọn Manage Jenkins sau đó chọn Credentials như hình



Vào credentials rồi bấm chọn vào 1 chữ system bất kì sau đó bấm “add domain”



- Ghi 1 tên domain tùy ý - “ưu tiên dễ nhớ mà làm cái khác nữa “ rồi bấm create
- Create xong rồi hiện ra trang đó luôn thì bấm Addcredentials chọn kind là username và password như hình rồi điền cái mình đăng ký này vào



- Kiểm tra kĩ lại rồi chọn create là xong
- Sau khi có ID credentials nhớ đổi trong jenkinsfile những thông tin của mình

Thế thôi là xong ; cpaj nhật xong thì đẩy code lên github check ở Jenkins thành công rồi vào xem dockerhub và docker có repo và images mới không là được

Cài làm với MinIO + dockedocker

Bước 1:

Tạo 1 file docker-compose.yml

```
version: '3.8'

services:
  minio:
    image: minio/minio:latest
    container_name: minio
    ports:
      - "9000:9000" # API
      - "9001:9001" # Web UI
    environment:
      MINIO_ROOT_USER: admin
      MINIO_ROOT_PASSWORD: password123
    command: server /data --console-address ":9001"
    volumes:
      - minio_data:/data

volumes:
  minio_data:
```

Sau đó chạy lệnh

Mở bash

```
docker compose up -d
```

Chạy thành công thì truy cập localhost:9001

BƯỚC 2: Tạo bucket trong MinIO

1. Truy cập `http://localhost:9001`
2. Đăng nhập bằng admin
3. Tạo bucket mới (ví dụ: `order-files`)

BƯỚC 3: TÍCH HỢP VỚI .NET CORE MVC

1. Cài NuGet:
Bash

```
dotnet add package AWSSDK.S3
```

2. Cấu hình appsettings.json:

```
"MinioSettings": {  
  "Endpoint": "http://localhost:9000",  
  "AccessKey": "admin",  
  "SecretKey": "password123",  
  "BucketName": "order-files"  
}
```

3. Tạo [MinioService.cs](#) ngay trong Service của dự án : Dùng `AmazonS3Client` để upload file

```
using Amazon.S3;  
using Amazon.S3.Transfer;  
using Amazon;  
  
public class MinioSettings  
{  
  public string Endpoint { get; set; }  
  public string AccessKey { get; set; }  
  public string SecretKey { get; set; }  
  public string BucketName { get; set; }  
}
```

```

public class MinioService
{
    private readonly IAmazonS3 _s3Client;
    private readonly string _bucketName;

    public MinioService(IConfiguration configuration)
    {
        var endpoint = configuration["MinioSettings:Endpoint"];
        var accessKey = configuration["MinioSettings:AccessKey"];
        var secretKey = configuration["MinioSettings:SecretKey"];
        _bucketName = configuration["MinioSettings:BucketName"];

        var config = new AmazonS3Config
        {
            ServiceURL = endpoint,
            ForcePathStyle = true // Cần thiết với MinIO
        };

        _s3Client = new AmazonS3Client(accessKey, secretKey, config);
    }

    public async Task UploadFileAsync(Stream fileStream, string fileName)
    {
        var fileTransfer = new TransferUtility(_s3Client);
        await fileTransfer.UploadAsync(fileStream, _bucketName, fileName);
    }
}

```

4. Tạo Controller và View cho người dùng upload file

Mở folder controller của dự án tạo luôn

[FileController.cs](#)

```

// minio controller ==> update minio to bucket cloud

using Microsoft.AspNetCore.Mvc;

public class FileController : Controller
{
    private readonly MinioService _minioService;

    public FileController(MinioService minioService)
    {
        _minioService = minioService;
    }

    [HttpGet]

```

```

public IActionResult Upload()
{
    return View(); // Trả về giao diện upload
}

[HttpPost]
public async Task<IActionResult> Upload(IFormFile file)
{
    if (file != null && file.Length > 0)
    {
        using var stream = file.OpenReadStream();
        await _minioService.UploadFileAsync(stream, file.FileName);
        ViewBag.Message = "Upload thành công!";
    }
    return View();
}
}

```

Tạo File/Upload.cshtml trong folder View của dự án gốc

```

<div
    style="max-width: 600px; margin: 40px auto; padding: 20px; background-color:
#1e1e2f; border-radius: 12px; box-shadow: 0 4px 10px rgba(0,0,0,0.3); color: #fff;">
    <h2 style="text-align:center; margin-bottom: 20px; color: #b3d4fc;">Upload file lên
MinIO</h2>

    <form asp-action="Upload" method="post" enctype="multipart/form-data">
        <div style="margin-bottom: 15px;">
            <label for="file">Chọn file để upload:</label><br />
            <input type="file" name="file" id="file" style="margin-top: 8px;
background-color: #fff; padding: 6px;" />
        </div>

        <button type="submit" style="
background-color: #4CAF50;
color: white;
padding: 10px 20px;
border: none;
border-radius: 6px;
cursor: pointer;
">Upload</button>
    </form>

    @if (ViewBag.Message != null)
    {
        <p style="margin-top: 20px; color: lightgreen; text-align:
center;">@ViewBag.Message</p>

```

```
}  
</div>
```

- Sau khi đẩy code lên thì mở web đã iis lên rồi thêm /File/Upload để mở trang có thể đẩy file lưu trên minio

Ví dụ: localhost:8089/File/Upload

Bước 4. TÍCH HỢP MINIO VÀO JENKINS CI/CD (Windows – bat)

1. Cài AWS CLI trên máy Jenkins

<https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

2. Dùng where aws để lấy đường dẫn aws.exe
3. Tạo demo 1 file build-log.txt để test sau muốn đđđâyfile khác thì thay tên

Rồi thêm vào file Jenkinsfile – cho ngay dưới sau đoạn code đẩy lên dockerhub nhé

```
//----- upload to minio -----  
  
stage('Tạo file test') {  
    steps {  
        bat 'echo Build thành công > build-log.txt'  
    }  
}  
  
stage('Cấu hình AWS CLI cho MinIO') {  
    steps {  
        bat '"C:\Program Files\Amazon\AWSCLIV2\aws.exe" configure set  
aws_access_key_id admin'  
        bat '"C:\Program Files\Amazon\AWSCLIV2\aws.exe" configure set  
aws_secret_access_key 12345678'  
        bat '"C:\Program Files\Amazon\AWSCLIV2\aws.exe" --endpoint-url  
http://localhost:9000 s3 cp build-log.txt s3://order-files/build-log.txt'  
    }  
}  
  
stage('Upload file lên MinIO') {  
    steps {  
        bat '"C:\Program Files\Amazon\AWSCLIV2\aws.exe" --endpoint-url  
http://localhost:9000 s3 cp build-log.txt s3://order-files/build-log.txt'  
    }  
}
```

Xong nhập lệnh để push lên github rồi vào xem jenkins chạy thành công thì mở minio ở localhost:9001 ra là có nhé

Cấu hình Grafana + docker

Bước 1: Thêm grafana vào docker-compose.yml

Cấu hình Traefik

*** TÓM TẮT CÁC BƯỚC TÍCH HỢP TRAEFIK (Reverse Proxy + Routing)

1. Cấu hình traefik service trong docker-compose.yml

```
traefik:
  image: traefik:v2.11
  container_name: traefik
  ports:
    - "80:80"      # HTTP reverse proxy
    - "8080:8080"  # Traefik dashboard
  command:
    - --api.insecure=true
    - --providers.docker=true
    - --providers.docker.exposedbydefault=false
    - --entrypoints.web.address=:80
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock:ro
  networks:
    - traefik-net
```

2. Yạo network dùng chung

```
networks:
  traefik-net:
    name: traefik-net
    driver: bridge
```

3. Gán các service (MinIO, Grafana, Web...) vào mạng **traefik-net** + thêm Traefik labels

```
services:
  traefik:
```


image: traefik:v2.11
container_name: traefik
ports:
- "80:80" # HTTP
- "8080:8080" # Traefik dashboard
command:
- --api.insecure=true
- --providers.docker=true
- --providers.docker.exposedbydefault=false
- --entrypoints.web.address=:80
labels:
- "traefik.enable=true"
- "traefik.http.routers.traefik.rule=Host(`traefik.localhost`)"
- "traefik.http.routers.traefik.entrypoints=web"
- "traefik.http.services.traefik.loadbalancer.server.port=8080"

volumes:
- /var/run/docker.sock:/var/run/docker.sock:ro

networks:
- traefik-net

sqlserver:
image: mcr.microsoft.com/mssql/server:2022-latest
container_name: sqlserver
ports:
- "1433:1433"

environment:
- ACCEPT_EULA=Y
- SA_PASSWORD=12345678
- MSSQL_PID=Express

volumes:
- sqlserver_data:/var/opt/mssql

networks:
- traefik-net

web-restaurant:
image: zyond/cicd:latest
container_name: web-restaurant
environment:
-

ConnectionStrings__DefaultConnection=Server=sqlserver,1433;Database=WebRestaurant12;
;User Id=sa;Password=12345678;

labels:
- "traefik.enable=true"
- "traefik.http.routers.webapp.rule=Host(`web.localhost`)"
- "traefik.http.services.webapp.loadbalancer.server.port=5000"

networks:
- traefik-net

grafana:
image: grafana/grafana:latest
container_name: grafana
volumes:
- grafana_data:/var/lib/grafana

labels:
- "traefik.enable=true"
- "traefik.http.routers.grafana.rule=Host(`grafana.localhost`)"
- "traefik.http.services.grafana.loadbalancer.server.port=3000"

networks:
- traefik-net

```

prometheus:
  image: prom/prometheus:latest
  container_name: prometheus
  volumes:
    - ./prometheus.yml:/etc/prometheus/prometheus.yml
  labels:
    - "traefik.enable=true"
    - "traefik.http.routers.prometheus.rule=Host(`prometheus.localhost`)"
    - "traefik.http.services.prometheus.loadbalancer.server.port=9090"
  networks:
    - traefik-net
minio:
  image: minio/minio:latest
  container_name: minio
  environment:
    MINIO_ROOT_USER: admin
    MINIO_ROOT_PASSWORD: 12345678
  command: server /data --console-address ":9001"
  volumes:
    - minio_data:/data
  labels:
    - "traefik.enable=true" # API MinIO (9000)
    - "traefik.http.routers.minio.rule=Host(`minio.localhost`)"
    - "traefik.http.routers.minio.entrypoints=web"
    - "traefik.http.routers.minio.service=minio-api"
    - "traefik.http.services.minio-api.loadbalancer.server.port=9000"
    # Console UI (9001)
    - "traefik.http.routers.minio-console.rule=Host(`minio-console.localhost`)"
    - "traefik.http.routers.minio-console.entrypoints=web"
    - "traefik.http.routers.minio-console.service=minio-console"
    - "traefik.http.services.minio-console.loadbalancer.server.port=9001"
  networks:
    - traefik-net
volumes:
  sqlserver_data:
  grafana_data:
  minio_data:
networks:
  traefik-net:
    name: traefik-net
    driver: bridge

```

Cập nhật đầy đủ rồi thì chạy lệnh bash : **[docker compose up -d](#)**

4. Truy cập dịch vụ qua domain *.**[localhost](#)**

Service	Truy cập URL
Traefik UI	http://traefik.localhost
MinIO API	http://minio.localhost
MinIO Console	http://minio-console.localhost

Grafana	http://grafana.localhost
Prometheus	http://prometheus.localhost