

**Implementation:** Using AWS cloud services to automate application deployment through a CI/CD pipeline.

**Objective:** To design and implement an automated CI/CD pipeline using Jenkins and AWS cloud services for efficient application deployment.

- Tools & Technologies Used

### AWS CodeBuild

AWS CodeBuild is a fully managed build service that compiles source code, runs tests, and produces build artifacts automatically without managing servers

### AWS CodeDeploy

AWS CodeDeploy is a deployment service that automates the process of deploying applications to EC2 instances, ensuring reliable and consistent releases.

### CI/CD Pipeline

A CI/CD pipeline is an automated workflow that enables continuous integration and continuous deployment by building, testing, and deploying applications whenever code changes are made.

### Jenkins

Jenkins is an open-source automation tool used to implement CI/CD pipelines by integrating source control, build, test, and deployment processes through plugins.

The screenshot shows the AWS EC2 'Launch an instance' wizard. In the 'Name and tags' step, the instance name is set to 'CICD-PROJECT-J'. In the 'Application and OS Images (Amazon Machine Image)' step, the operating system is selected as 'Ubuntu' from the 'Quick Start' options. The summary on the right indicates 1 instance, using the Canonical, Ubuntu, 24.04, amd64 AMI (ami-0ecb62995f68bb549), with a t3.micro instance type, a new security group, and 1 volume(s) - 8 GiB storage. The 'Launch instance' button is highlighted.

EC2 > Instances > Launch an instance

## Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

### Name and tags Info

Name

CICD-PROJECT-J

Add additional tags

### Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Search our full catalog including 1000s of application and OS images

#### Quick Start

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux Debian

>

[Browse more AMIs](#)  
Including AMIs from AWS, Marketplace and the Community

### Summary

Number of instances Info

1

Software Image (AMI)

Canonical, Ubuntu, 24.04, amd64... [read more](#)  
ami-0ecb62995f68bb549

Virtual server type (instance type)

t3.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

[Cancel](#) [Launch instance](#) [Preview code](#)

I'm starting the process of creating an AWS EC2 virtual server. I navigate to EC2 > Instances > Launch an instance and name my instance "CICD-PROJECT-J" for easy identification. I select Ubuntu 24.04 from the Quick Start options as my operating system because it's reliable for web development. I choose the t3.



Search

[Alt+S]



United States (N. Virginia) ▾

jay2025 (1827-1758-6119) ▾  
jay2025

EC2 &gt; Instances &gt; Launch an instance

i

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t3.micro

Family: t3 2 vCPU 1 GiB Memory Current generation: true  
On-Demand Ubuntu Pro base pricing: 0.0139 USD per Hour  
On-Demand SUSE base pricing: 0.0104 USD per Hour On-Demand Linux  
On-Demand RHEL base pricing: 0.0392 USD per Hour On-Demand Wind

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure

Key pair name - required

Select

▼ Network settings [Info](#)

Network [Info](#)  
vpc-098922d693452fb25

Subnet [Info](#)  
No preference (Default subnet in any availability zone)

Create instance

Create key pair

▼ Create key pair

Key pair name

Key pairs allow you to connect to your instance securely.

PROJECT-02

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

RSA RSA encrypted private and public key pair

ED25519 ED25519 encrypted private and public key pair

Private key file format

.pem For use with OpenSSH

.ppk For use with PuTTY

When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more ↗](#)

Cancel Create key pair

Summary

Number of instances [Info](#)

Software Image (AMI)

Canonical, Ubuntu, 24.04, amd64... [read more](#)  
0ecb62995f68bb549

Virtual server type (instance type)

micro

Network (security group)

Volume (volumes)

Launch instance

Here I'm creating a security key pair that will allow me to connect to my server remotely. I click "Create key pair" and name it "PROJECT-02" to match my project. I select RSA encryption type and .pem file format because it works with OpenSSH on my Windows/Linux system. Once I click "Create key pair," the private key file downloads to my computer - I need to keep this safe as it's my only way to access the server via SSH.

The screenshot shows the AWS EC2 Instances Launch Wizard. On the left, under 'Firewall (security groups)', the 'Create security group' option is selected. A security group named 'MY-WEB-SG' is being created, with a description of 'launch-wizard-1 created 2026-01-13T10:02:36.838Z'. In the 'Inbound Security Group Rules' section, there is one rule: 'Security group rule 1 (TCP, 22, 49.36.83.23/32, SSH ACCESS)'. This rule uses TCP protocol on port 22, source type 'My IP' (49.36.83.23/32), and has a description 'SSH ACCESS'. On the right, the 'Summary' section shows 1 instance being launched, using the Canonical, Ubuntu 24.04 AMI, t3.micro instance type, and a new security group. It also lists 1 volume (8 GiB). At the bottom right are 'Cancel', 'Launch instance', and 'Preview code' buttons.

Now I'm setting up firewall rules by creating a new security group named "MY-WEB-SG". The first rule I add is for SSH access on port 22 using TCP protocol, which allows me to remotely manage the server through terminal. I set the source to "My IP" and enter my current IP address (49.36.83.23/32) so only I can SSH into the server, blocking unauthorized access attempts. I label this rule "SSH ACCESS" for documentation purposes.

The screenshot shows the AWS EC2 'Launch an instance' wizard. On the left, three security group rules are defined:

- Security group rule 2 (TCP, 80, 0.0.0.0/0, FOR Web traffic)**: Type: HTTP, Protocol: TCP, Port range: 80, Source type: Anywhere, Description: FOR Web traffic.
- Security group rule 3 (TCP, 8080, 0.0.0.0/0, Jenkins)**: Type: Custom TCP, Protocol: TCP, Port range: 8080, Source type: Anywhere, Description: Jenkins.
- Security group rule 4 (TCP, 0)**: Type: Custom TCP, Protocol: TCP, Port range: 0, Source type: Anywhere.

On the right, the instance configuration summary is shown:

- Number of instances**: 1
- Software Image (AMI)**: Canonical, Ubuntu, 24.04, amd64... (read more)
- Virtual server type (instance type)**: t3.micro
- Firewall (security group)**: New security group
- Storage (volumes)**: 1 volume(s) - 8 GiB

At the bottom right are 'Cancel', 'Launch instance' (highlighted in orange), and 'Preview code' buttons.

I'm adding more firewall rules to allow different types of traffic to my server. I create a rule for HTTP on port 80 with source "Anywhere" (0.0.0.0/0) and label it "FOR Web traffic" - this lets anyone access my website through a browser. Next, I add port 8080 using Custom TCP for Jenkins, also set to "Anywhere" so the Jenkins CI/CD dashboard can be accessed publicly. These rules enable my web application and automation tools to be reached from the internet.

The screenshot shows the AWS EC2 'Launch an instance' wizard. In the top left, the navigation path is 'EC2 > Instances > Launch an instance'. The top right shows the user 'jay2025 (1827-1758-6119)' and the region 'United States (N. Virginia)'. The main area is divided into two sections: 'Security group rule 4' and 'Configure storage'.

**Security group rule 4 (TCP, 3000, 0.0.0.0/0, Node.js app)**

- Type**: Custom TCP
- Protocol**: TCP
- Port range**: 3000
- Description - optional**: Node.js app

A warning message in a yellow box states: "⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only." with a close button 'X'.

**Add security group rule**

**Configure storage**

- Root volume: 20 GiB gp3, 3000 IOPS, Not encrypted
- Advanced** link

**Add new volume**

A note at the bottom says: "The selected AMI contains instance store volumes, however the instance does not allow any instance store volumes. None of the instance store volumes will be used."

**Summary**

- Number of instances**: 1
- Software Image (AMI)**: Canonical, Ubuntu, 24.04, amd64... [read more](#) ami-0ecb62995f68bb549
- Virtual server type (instance type)**: t3.micro
- Firewall (security group)**: New security group
- Storage (volumes)**: 1 volume(s) - 20 GiB

**Launch instance** button, **Preview code** button, and **Cancel** button.

I'm configuring the fourth security rule for port 3000 with Custom TCP protocol and "Anywhere" source, which I describe as "Node.js app". This opens the port commonly used by Node.js applications like Express or React dev servers. AWS shows me a warning that 0.0.0.0/0 allows

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\JAY> cd download
cd : Cannot find path 'C:\Users\JAY\download' because it does not exist.
At line:1 char:1
+ cd download
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (C:\Users\JAY\download:String) [Set-Location], ItemNotFoundException
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.SetLocationCommand

PS C:\Users\JAY> cd downloads
PS C:\Users\JAY\downloads> ssh -i "PROJECT-02.pem" ubuntu@ec2-54-226-136-58.compute-1.amazonaws.com
The authenticity of host 'ec2-54-226-136-58.compute-1.amazonaws.com (54.226.136.58)' can't be established.
ED25519 key fingerprint is SHA256:9lu6CLhLhJ1bL9tcTSMPS40ugp6/QpPzXyK+XfZW9Uc.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-226-136-58.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Tue Jan 13 10:12:53 UTC 2026

System load:  0.9              Temperature:        -273.1 C
Usage of /:   9.4% of 18.33GB  Processes:          117
```

Now connected to powershell

```
ubuntu@ip-172-31-20-6:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1693 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [313 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [175 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [15.9 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1510 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [307 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [377 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [31.4 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [2426 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [554 kB]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 c-n-f Metadata [516 B]
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [30.3 kB]
Get:26 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [6048 B]
Get:27 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:28 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [488 B]
Get:29 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Packages [40.4 kB]
```

**sudo apt update** I'm updating the package list from Ubuntu repositories to get the latest information about available packages and their versions. This ensures I can install the most recent stable software versions.

```
ubuntu@ip-172-31-20-6: ~      x + 
82 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-20-6:~$ sudo apt upgrade -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following NEW packages will be installed:
  linux-aws-6.14.0-1018 linux-aws-6.14-tools-6.14.0-1018 linux-headers-6.14.0-1018-aws
    linux-image-6.14.0-1018-aws linux-modules-6.14.0-1018-aws linux-tools-6.14.0-1018-aws
The following packages will be upgraded:
  apparmor bind9-dnsutils bind9-host bind9-libs bsddextrautils bsduutils dhcpcd-base dirmngr eject fdisk gir1.2-glib-2.0
  gir1.2-packagekitglib-1.0 gnupg gnupg-l10n gnupg-utils gpg gpg-agent gpg-wks-client gpgconf gpgsm gpgv
  intel-microcode keyboxd landscape-common libapparmor1 libblkid1 libdrm-common libdrm2 libfdisk1 libglib2.0-0t64
  libglib2.0-bin libglib2.0-data libmbim-glib4 libmbim-proxy libmbim-utils libmount1 libnetplan1 libnss-systemd
  libpackagekit-glib2-18 libpam-systemd libpng16-16t64 libpython3-stdlib libpython3.12-minimal libpython3.12-stdlib
  libpython3.12t64 libsmartcols1 libsodium23 libssh-4 libsystemd-shared libsystemd0 libtasn1-6 libudev1 libuuid1
  libxml2 libxslt1.1 linux-aws linux-headers-aws linux-image-aws linux-tools-common mount netplan-generator netplan.io
  packagekit packagekit-tools python-apt-common python3 python3-apt python3-minimal python3-netplan python3-urllib3
  python3.12 python3.12-minimal snapd systemd systemd-dev systemd-resolved systemd-sysv ubuntu-pro-client
  ubuntu-pro-client-l10n udev util-linux uuid-runtime
82 upgraded, 6 newly installed, 0 to remove and 0 not upgraded.
40 standard LTS security updates
Need to get 149 MB of archives.
After this operation, 179 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 bsduutils amd64 1:2.39.3-9ubuntu6.4 [95.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 util-linux amd64 2.39.3-9ubuntu6.4 [1128 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 mount amd64 2.39.3-9ubuntu6.4 [118 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 ubuntu-pro-client-l10n amd64 37.1ubuntu0~2
```

**sudo apt upgrade -y** I'm upgrading all currently installed packages to their latest versions using the -y flag for automatic yes to prompts. This keeps my system secure and up-to-date with bug fixes and improvements.

```
ubuntu@ip-172-31-20-6:~ x + v
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-20-6:~$ curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash
2026-01-13 10:18:19 - Installing pre-requisites
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.6).
curl set to manually installed.
gnupg is already the newest version (2.4.4-2ubuntu17.4).
gnupg set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 3970 B of archives.
After this operation, 36.9 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 apt-transport-https all 2.8.3 [3970 B]
Fetched 3970 B in 0s (269 kB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 103390 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.8.3_all.deb ...
```

**[curl -fsSL https://deb.nodesource.com/setup\\_20.x | sudo -E bash](https://deb.nodesource.com/setup_20.x)** - I'm downloading and executing the NodeSource setup script to add the Node.js 20.x repository to my system. This allows me to install the latest LTS version of Node.js instead of the older version in Ubuntu's default repositories.

```
ubuntu@ip-172-31-20-6:~      + |   
2026-01-13 10:18:38 - To install N|solid Runtime, run: apt install nsolid -y  
  
ubuntu@ip-172-31-20-6:~$ sudo apt install -y nodejs  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following NEW packages will be installed:  
  nodejs  
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.  
Need to get 32.0 MB of archives.  
After this operation, 197 MB of additional disk space will be used.  
Get:1 https://deb.nodesource.com/node_20.x nodistro/main amd64 nodejs amd64 20.19.6-1nodesource1 [32.0 MB]  
Fetched 32.0 MB in 0s (75.5 MB/s)  
Selecting previously unselected package nodejs.  
(Reading database ... 103394 files and directories currently installed.)  
Preparing to unpack .../nodejs_20.19.6-1nodesource1_amd64.deb ...  
Unpacking nodejs (20.19.6-1nodesource1) ...  
Setting up nodejs (20.19.6-1nodesource1) ...  
Processing triggers for man-db (2.12.0-4build2) ...  
Scanning processes...  
Scanning candidates...  
Scanning linux images...  
  
Pending kernel upgrade!  
Running kernel version:  
  6.14.0-1015-aws  
Diagnostics:  
  The currently running kernel version is not the expected kernel version 6.14.0-1018-aws.  
  
Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.
```

**sudo apt install -y nodejs** I'm installing Node.js (which includes npm - Node Package Manager) from the NodeSource repository. Node.js is required to run my FastFood web application backend server.

```
ubuntu@ip-172-31-20-6:~ + - X
Pending kernel upgrade!
Running kernel version:
  6.14.0-1015-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.14.0-1018-aws.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

Restarting services...

Service restarts being deferred:
/etc/needrestart/restart.d/dbus.service
systemctl restart getty@tty1.service
systemctl restart networkd-dispatcher.service
systemctl restart serial-getty@ttyS0.service
systemctl restart systemd-logind.service
systemctl restart unattended-upgrades.service

No containers need to be restarted.

User sessions running outdated binaries:
ubuntu @ session #1: sshd[1083]
ubuntu @ user manager service: systemd[1088]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-20-6:~$ node --version
v20.19.6
ubuntu@ip-172-31-20-6:~$ npm --version
10.8.2
ubuntu@ip-172-31-20-6:~$
```

**node --version** I'm verifying that Node.js installed correctly by checking its version number. This confirms the installation was successful and shows which version is now available on my system.

**npm --version** I'm checking the npm (Node Package Manager) version to confirm it was installed along with Node.js. npm is essential for managing JavaScript dependencies in my project.

```
ubuntu@ip-172-31-20-6:~$ sudo npm install -g pm2
```

```
added 133 packages in 8s
```

```
13 packages are looking for funding
```

```
  run `npm fund` for details
```

```
npm notice
```

```
npm notice New major version of npm available! 10.8.2 -> 11.7.0
```

```
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.7.0
```

```
npm notice To update run: npm install -g npm@11.7.0
```

```
npm notice
```

```
ubuntu@ip-172-31-20-6:~$
```

**sudo npm install -g pm2** I'm installing PM2 globally using the -g flag, which is a production-grade process manager for Node.js applications. PM2 keeps my app running continuously, restarts it if it crashes, and manages logs efficiently.

**pm2 --version** I'm verifying PM2 installation by checking its version number to ensure the global installation completed successfully and PM2 commands are available system-wide.

```
ubuntu@ip-172-31-20-6:~$ sudo env PATH=$PATH:/usr/bin /usr/lib/node_modules/pm2/bin/pm2 startup systemd -u ubuntu --hp /home/ubuntu
```



Runtime Edition

PM2 is a Production Process Manager for Node.js applications

I'm executing the PM2-generated startup command to configure systemd to automatically launch PM2 (and all saved processes) when the system boots. This ensures my application starts automatically after server restarts without manual intervention.

```
ubuntu@ip-172-31-20-6:~$ pm2 save  
[PM2] Saving current process list...  
[PM2][WARN] PM2 is not managing any process, skipping save...  
[PM2][WARN] To force saving use: pm2 save --force  
ubuntu@ip-172-31-20-6:~$
```

**pm2 save** I'm saving the current list of PM2-managed processes to disk so they can be automatically resurrected on system reboot. Any apps currently running under PM2 will restart automatically after a reboot.

```
[PM2][WARN] PM2 is not managing any process, skipping save...
[PM2][WARN] To force saving use: pm2 save --force
ubuntu@ip-172-31-20-6:~$ sudo apt install -y fontconfig openjdk-17-jre
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
adwaita-icon-theme alsamixer alsavu alsound2-data alsound2t64 alsound2t64-alsa-ucm-conf alsound2t64-at-spi2-common alsound2t64-at-spi2-core alsound2t64-ca-certificates-java alsound2t64-dconf-gsettings-backend alsound2t64-dconf-service alsound2t64-fontconfig alsound2t64-fontconfig-config alsound2t64-fonts-dejavu-core alsound2t64-fonts-dejavu-extra alsound2t64-fonts-dejavu-mono alsound2t64-gsettings-desktop-schemas alsound2t64-gtk-update-icon-cache alsound2t64-hicolor-icon-theme alsound2t64-humanity-icon-theme alsound2t64-java-common alsound2t64-libasound2-data alsound2t64-libatk-bridge2.0-0t64 alsound2t64-libatk-wrapper-java alsound2t64-libatk-wrapper-jni alsound2t64-libatk1.0-0t64 alsound2t64-libatspi2.0-0t64 alsound2t64-libavahi-client3 alsound2t64-libavahi-common-data alsound2t64-libavahi-common3 alsound2t64-libcairo-gobject2 alsound2t64-libcairo2 alsound2t64-libcups2t64 alsound2t64-libdatrie1 alsound2t64-libdconf1 alsound2t64-libdeflate0 alsound2t64-libdrm-amdgpu1 alsound2t64-libdrm-intel1 alsound2t64-libfontconfig1 alsound2t64-libgail-common alsound2t64-libgail18t64 alsound2t64-libgbm1 alsound2t64-libgdk-pixbuf-2.0-0 alsound2t64-libgdk-pixbuf2.0-bin alsound2t64-libgdk-pixbuf2.0-common alsound2t64-libgif7 alsound2t64-libgl1 alsound2t64-libgl1-mesa-dri alsound2t64-libglvnd0 alsound2t64-libglx-mesa0 alsound2t64-libglx0 alsound2t64-libgraphite2-3 alsound2t64-libgtk2.0-0t64 alsound2t64-libgtk2.0-bin alsound2t64-libgtk2.0-common alsound2t64-libharfbuzz0b alsound2t64-libice6 alsound2t64-libjbig0 alsound2t64-libjpeg-turbo8 alsound2t64-libjpeg8 alsound2t64-liblcms2-2 alsound2t64-liblerc4 alsound2t64-libllvm20 alsound2t64-libpango-1.0-0 alsound2t64-libpangocairo-1.0-0 alsound2t64-libpangoft2-1.0-0 alsound2t64-libpciaaccess0 alsound2t64-libpcsclite1 alsound2t64-libpixman-1-0 alsound2t64-librsvg2-2 alsound2t64-librsvg2-common alsound2t64-libsharpyuv0 alsound2t64-libsm6 alsound2t64-libthai-data alsound2t64-libthai0 alsound2t64-libtiff6 alsound2t64-libvulkan1 alsound2t64-libwayland-client0 alsound2t64-libwayland-server0 alsound2t64-libwebp7 alsound2t64-libx11-xcb1 alsound2t64-libxaw7 alsound2t64-libxcb-dri3-0 alsound2t64-libxcb-glx0 alsound2t64-libxcb-present0 alsound2t64-libxcb-randr0 alsound2t64-libxcb-render0 alsound2t64-libxcb-shape0 alsound2t64-libxcb-shm0 alsound2t64-libxcb-sync1 alsound2t64-libxcb-xfixes0 alsound2t64-libxcomposite1 alsound2t64-libxcursor1 alsound2t64-libxdamage1 alsound2t64-libxfixes3 alsound2t64-libxft2 alsound2t64-libxi6 alsound2t64-libxinerama1 alsound2t64-libxkbfile1 alsound2t64-libxmu6 alsound2t64-libxpm4 alsound2t64-libxrandr2 alsound2t64-libxrender1 alsound2t64-libxshmfence1 alsound2t64-libxt6t64 alsound2t64-libxtst6 alsound2t64-libxv1 alsound2t64-libxxf86dga1 alsound2t64-libxxf86vm1 mesa-libgallium mesa-vulkan-drivers openjdk-17-jre-headless session-migration ubuntu-mono x11-common x11-utils
Suggested packages:
```

**sudo apt install -y fontconfig openjdk-17-jre** I'm installing Java 17 JRE (Java Runtime Environment) along with fontconfig, which are prerequisites for running Jenkins. Jenkins is a Java-based application that requires JRE to function.

```
ubuntu@ip-172-31-20-6:~$ java --version
openjdk 17.0.17 2025-10-21
OpenJDK Runtime Environment (build 17.0.17+10-Ubuntu-124.04)
OpenJDK 64-Bit Server VM (build 17.0.17+10-Ubuntu-124.04, mixed mode, sharing)
ubuntu@ip-172-31-20-6:~$
```

**java --version** I'm verifying the Java installation and checking which version is installed to confirm Java 17 is available and properly configured on my system.

```
ubuntu@ip-172-31-20-6:~$ java --version
openjdk 17.0.17 2025-10-21
OpenJDK Runtime Environment (build 17.0.17+10-Ubuntu-124.04)
OpenJDK 64-Bit Server VM (build 17.0.17+10-Ubuntu-124.04, mixed mode, sharing)
ubuntu@ip-172-31-20-6:~$ curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
ubuntu@ip-172-31-20-6:~$ echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list
deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/
ubuntu@ip-172-31-20-6:~$ |
```

**curl -fsSL <https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key> | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null**

I'm downloading the official Jenkins GPG key and adding it to my system's keyring for package verification. This ensures that only authentic Jenkins packages from the official repository can be installed.

**cho "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] <https://pkg.jenkins.io/debian-stable> binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list**

I'm adding the Jenkins stable repository to my apt sources list with proper GPG key verification. This tells apt where to download Jenkins packages from and how to verify their authenticity.

```
ubuntu@ip-172-31-20-6:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Ign:5 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:6 https://pkg.jenkins.io/debian-stable binary/ Release [2044 B]
Get:7 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Hit:8 https://deb.nodesource.com/node_20.x nodistro InRelease
Get:9 https://pkg.jenkins.io/debian-stable binary/ Packages [30.3 kB]
Fetched 33.2 kB in 1s (65.3 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
ubuntu@ip-172-31-20-6:~$
```

**sudo apt update** I'm updating the package list again to include packages from the newly added Jenkins repository. This allows apt to see and install Jenkins from the official Jenkins repository.

```
ubuntu@ip-172-31-20-6:~$ sudo apt install jenkins -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  net-tools
The following NEW packages will be installed:
  jenkins net-tools
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 95.2 MB of archives.
After this operation, 96.3 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 net-tools amd64 2.10-0.1ubuntu4.4 [204 kB]
Get:2 https://pkg.jenkins.io/debian-stable binary/ jenkins 2.528.3 [95.0 MB]
Fetched 95.2 MB in 2s (62.1 MB/s)
Selecting previously unselected package net-tools.
(Reading database ... 123477 files and directories currently installed.)
Preparing to unpack .../net-tools_2.10-0.1ubuntu4.4_amd64.deb ...
Unpacking net-tools (2.10-0.1ubuntu4.4) ...
Selecting previously unselected package jenkins.
Preparing to unpack .../jenkins_2.528.3_all.deb ...
Unpacking jenkins (2.528.3) ...
Setting up net-tools (2.10-0.1ubuntu4.4) ...
Setting up jenkins (2.528.3) ...
```

**sudo apt install jenkins -y** I'm installing Jenkins CI/CD automation server which will handle building, testing, and deploying my FastFood application automatically whenever I push code changes to GitHub.

```
ubuntu@ip-172-31-20-6:~$ sudo systemctl start jenkins
ubuntu@ip-172-31-20-6:~$ sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
ubuntu@ip-172-31-20-6:~$
```

**sudo systemctl start jenkins** I'm starting the Jenkins service for the first time. This launches the Jenkins web server which will be accessible on port 8080 for configuration and creating CI/CD pipelines.

**sudo systemctl enable jenkins** I'm configuring Jenkins to start automatically when the system boots. This ensures Jenkins is always available after server restarts without manual intervention.

```
ubuntu@ip-172-31-20-6:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
  Active: active (running) since Tue 2026-01-13 10:28:16 UTC; 1min 38s ago
    Main PID: 16570 (java)
       Tasks: 38 (limit: 1008)
      Memory: 304.8M (peak: 322.4M)
        CPU: 24.529s
      CGroup: /system.slice/jenkins.service
              └─16570 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenki>

Jan 13 10:28:10 ip-172-31-20-6 jenkins[16570]: [LF]> This may also be found at: /var/lib/jenkins/secrets/initialAdminPa>
Jan 13 10:28:10 ip-172-31-20-6 jenkins[16570]: [LF]>
Jan 13 10:28:10 ip-172-31-20-6 jenkins[16570]: [LF]> ****
Jan 13 10:28:10 ip-172-31-20-6 jenkins[16570]: [LF]> ****
Jan 13 10:28:10 ip-172-31-20-6 jenkins[16570]: [LF]> ****
Jan 13 10:28:15 ip-172-31-20-6 jenkins[16570]: 2026-01-13 10:28:15.976+0000 [id=32]           INFO      jenkins.InitReac>
Jan 13 10:28:16 ip-172-31-20-6 jenkins[16570]: 2026-01-13 10:28:16.073+0000 [id=23]           INFO      hudson.lifecycle>
Jan 13 10:28:16 ip-172-31-20-6 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Jan 13 10:28:16 ip-172-31-20-6 jenkins[16570]: 2026-01-13 10:28:16.155+0000 [id=49]           INFO      h.m.DownloadServ>
Jan 13 10:28:16 ip-172-31-20-6 jenkins[16570]: 2026-01-13 10:28:16.155+0000 [id=49]           INFO      hudson.util.Retri>
Lines 1-20/20 (END)
```

**sudo systemctl status jenkins** I'm checking the Jenkins service status to verify it's running properly and to see its current state, PID, and any recent log messages.

```
ubuntu@ip-172-31-20-6:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
43de89ac2c3740f9a39b13759bd09d86  
ubuntu@ip-172-31-20-6:~$ sudo apt install -y ruby wget  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
wget is already the newest version (1.21.4-1ubuntu4.1).  
wget set to manually installed.  
The following additional packages will be installed:  
  fonts-lato javascript-common libjs-jquery libruby libruby3.2 rake ruby-net-telnet ruby-rubygems ruby-sdbm  
    ruby-webrick ruby-xmlrpc ruby3.2 rubygems-integration unzip zip  
Suggested packages:  
  apache2 | lighttpd | httpd ri ruby-dev bundler  
The following NEW packages will be installed:  
  fonts-lato javascript-common libjs-jquery libruby libruby3.2 rake ruby ruby-net-telnet ruby-rubygems ruby-sdbm  
    ruby-webrick ruby-xmlrpc ruby3.2 rubygems-integration unzip zip  
0 upgraded, 16 newly installed, 0 to remove and 0 not upgraded.  
Need to get 9277 kB of archives.  
After this operation, 42.0 MB of additional disk space will be used.  
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 fonts-lato all 2.015-1 [2781 kB]  
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 javascript-common all 11+nmu1 [5936 B]  
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libjs-jquery all 3.6.1+dfsg+~3.5.14-1 [328 kB]  
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 rubygems-integration all 1.18 [5336 B]  
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 ruby3.2 amd64 3.2.3-1ubuntu0.24.04.6 [50.7 kB]  
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 ruby-rubygems all 3.4.20-1 [238 kB]  
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 ruby amd64 1:3.2~ubuntu1 [3466 B]  
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 ruby3.2 amd64 3.2.3-1ubuntu0.24.04.6 [61.6 kB]
```

**sudo cat /var/lib/jenkins/secrets/initialAdminPassword** I'm displaying the initial admin password that Jenkins generated during installation. I need this password to complete the Jenkins setup wizard on first login through the web interface at port 8080.

**sudo apt install -y ruby wget** I'm installing Ruby and wget which are dependencies required by the AWS CodeDeploy agent installer script. Ruby is needed to run the agent, and wget is used to download files.

```
ubuntu@ip-172-31-20-6:/$ cd /tmp
ubuntu@ip-172-31-20-6:/tmp$ wget https://aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com/latest/install
--2026-01-13 10:36:25--  https://aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com/latest/install
Resolving aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com (aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com)... 5
2.217.136.74, 54.231.236.122, 16.15.223.241, ...
Connecting to aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com (aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com)|52.217.136.74|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19045 (19K) []
Saving to: 'install'

install          100%[=====] 18.60K --.-KB/s   in 0.001s

2026-01-13 10:36:25 (24.5 MB/s) - 'install' saved [19045/19045]

ubuntu@ip-172-31-20-6:/tmp$
```

**cd /tmp** I'm changing to the temporary directory where I'll download the CodeDeploy agent installer. The /tmp directory is appropriate for temporary downloads that don't need to be preserved.

```
ubuntu@ip-172-31-20-6:/tmp$ chmod +x ./install
ubuntu@ip-172-31-20-6:/tmp$ sudo ./install auto
I, [2026-01-13T10:37:08.224491 #17227] INFO -- : Starting Ruby version check.
W, [2026-01-13T10:37:08.224633 #17227] WARN -- : The Ruby version in /usr/bin/ruby3.2 is 3.2.3, . Attempting to install anyway.
I, [2026-01-13T10:37:08.224672 #17227] INFO -- : Starting update check.
I, [2026-01-13T10:37:08.224690 #17227] INFO -- : Attempting to automatically detect supported package manager type for system...
W, [2026-01-13T10:37:08.240523 #17227] WARN -- : apt-get found but no gdebi. Installing gdebi with `apt-get install gdebi-core -y`...
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  gdebi-core
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 132 kB of archives.
After this operation, 861 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 gdebi-core all 0.9.5.7+nmu7 [132 kB]
Fetched 132 kB in 0s (6706 kB/s)
Selecting previously unselected package gdebi-core.
(Reading database ... 126657 files and directories currently installed.)
Preparing to unpack .../gdebi-core_0.9.5.7+nmu7_all.deb ...
Unpacking gdebi-core (0.9.5.7+nmu7) ...
Setting up gdebi-core (0.9.5.7+nmu7) ...
/usr/share/gdebi/GDebi/GDebiCli.py:159: SyntaxWarning: invalid escape sequence '\S'
```

**chmod +x ./install** I'm making the downloaded install script executable by adding execute permissions. This allows me to run the script as a program to install the CodeDeploy agent.

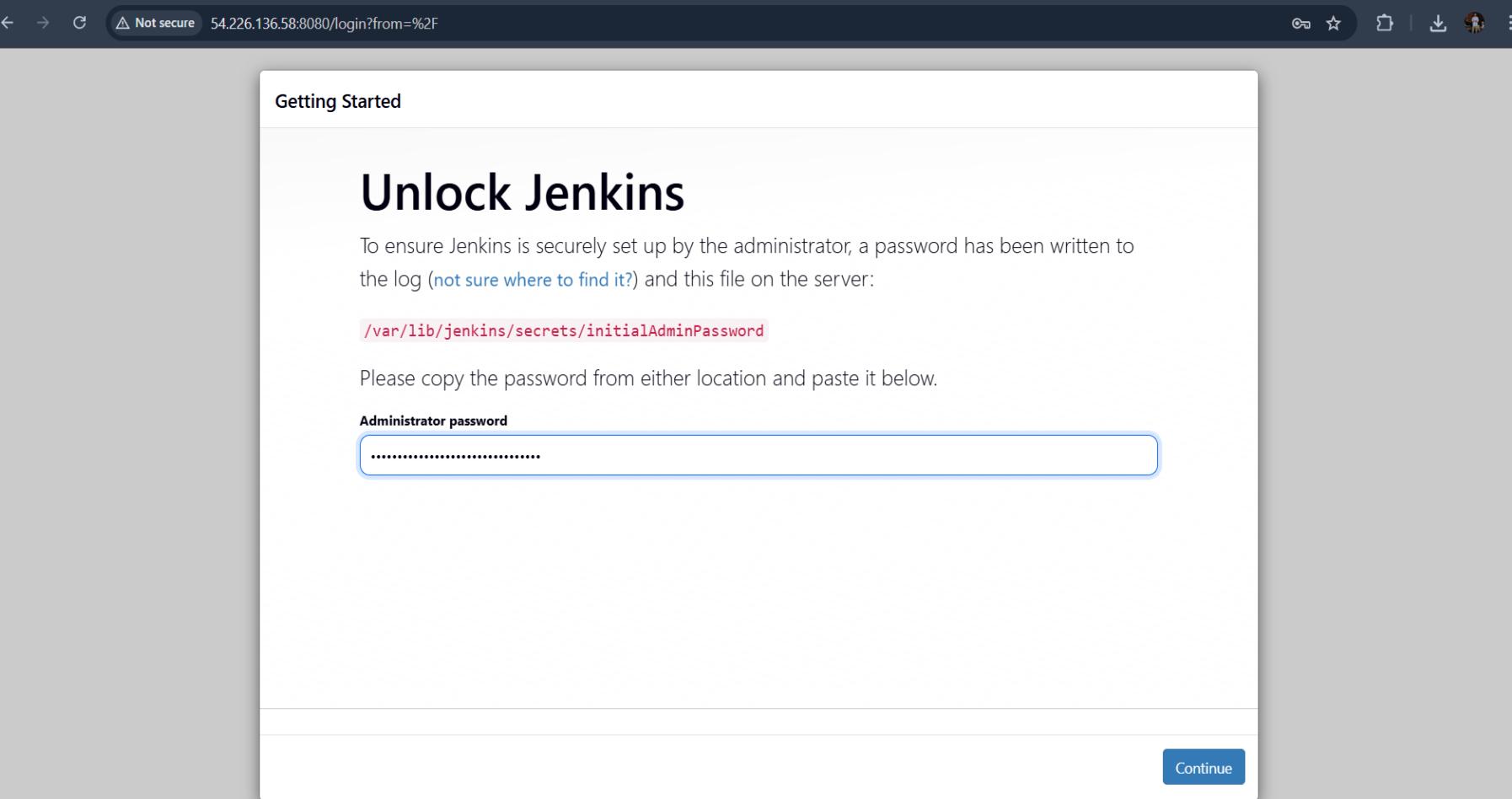
**sudo ./install auto** I'm running the CodeDeploy agent installer with the auto parameter to automatically install and configure the agent. The agent will listen for deployment commands from AWS CodeDeploy service.

```
ubuntu@ip-172-31-20-6:/tmp$ sudo service codedeploy-agent start
ubuntu@ip-172-31-20-6:/tmp$ sudo service codedeploy-agent status
● codedeploy-agent.service - LSB: AWS CodeDeploy Host Agent
    Loaded: loaded (/etc/init.d/codedeploy-agent; generated)
    Active: active (running) since Tue 2026-01-13 10:37:20 UTC; 40s ago
      Docs: man:systemd-sysv-generator(8)
   Process: 17533 ExecStart=/etc/init.d/codedeploy-agent start (code=exited, status=0/SUCCESS)
     Tasks: 3 (limit: 1008)
    Memory: 68.7M (peak: 68.8M)
       CPU: 1.177s
      CGroup: /system.slice/codedeploy-agent.service
              └─17539 "codedeploy-agent: master 17539"
                  ├─17542 "codedeploy-agent: InstanceAgent::Plugins::CodeDeployPlugin::CommandPoller of master 17539"
                  └─17543 "codedeploy-agent: InstanceAgent::Plugins::CodeDeployPlugin::CommandPoller of master 17539"

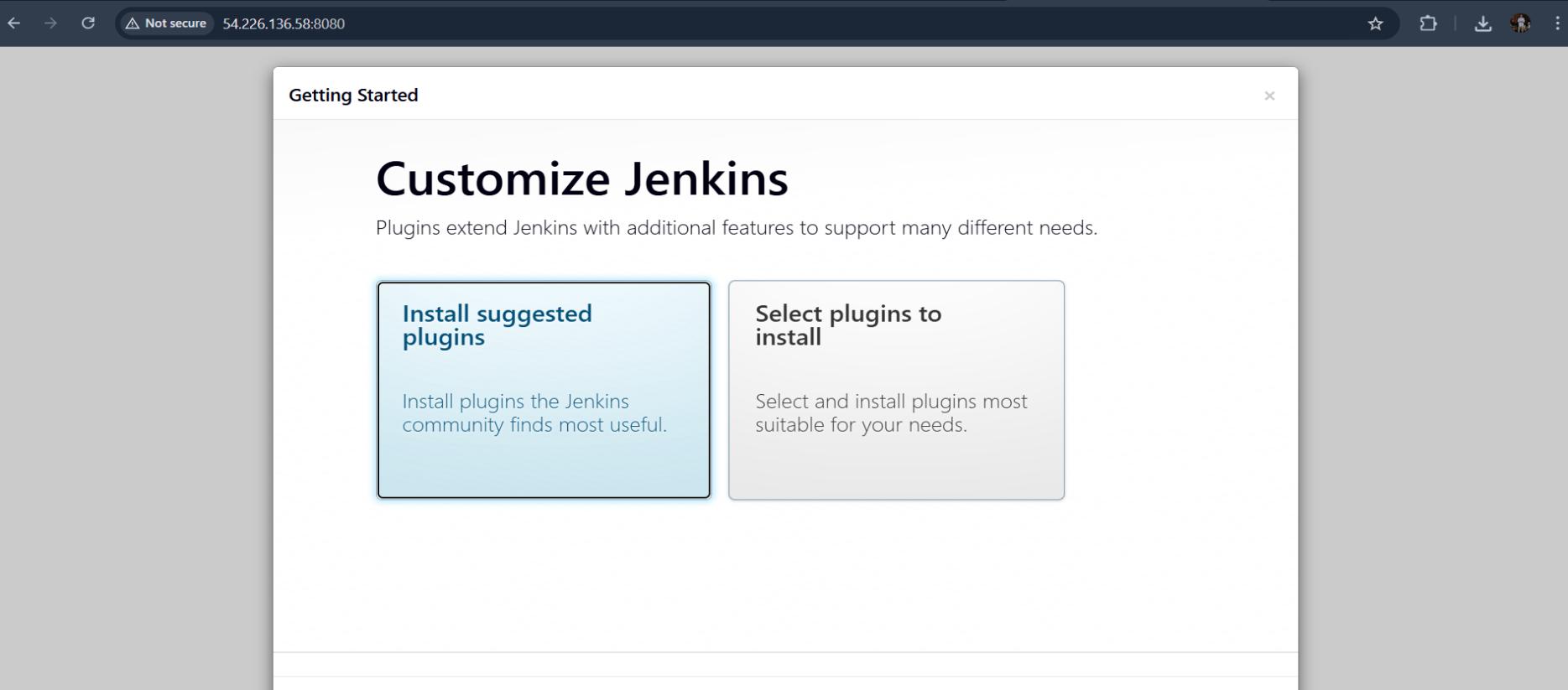
Jan 13 10:37:19 ip-172-31-20-6 systemd[1]: Starting codedeploy-agent.service - LSB: AWS CodeDeploy Host Agent...
Jan 13 10:37:20 ip-172-31-20-6 codedeploy-agent[17533]: Starting codedeploy-agent:
Jan 13 10:37:20 ip-172-31-20-6 systemd[1]: Started codedeploy-agent.service - LSB: AWS CodeDeploy Host Agent.
ubuntu@ip-172-31-20-6:/tmp$
```

**sudo service codedeploy-agent start** I'm starting the CodeDeploy agent service using the legacy service command. This launches the agent so it can receive and execute deployment instructions from AWS CodeDeploy.

**sudo service codedeploy-agent status** I'm checking the CodeDeploy agent status using the legacy service command to verify it's running and ready to handle deployments.



- This screen confirms Jenkins is successfully installed and running on your EC2 instance.
- It prompts for the initial admin password stored at .  
`/var/lib/Jenkins/secreats/initialAdminpassword`
- This step ensures only the EC2 administrator can complete Jenkins setup securely.
- Once entered, you'll proceed to plugin installation and user creation.



- The “Install suggested plugins” option is ideal for beginners—it includes commonly used tools like Git, Pipeline, and SSH.
- The “Select plugins to install” option lets you manually choose plugins based on my project needs.
- This step sets up Jenkins for CI/CD tasks like code integration, testing, and deployment.

## Getting Started

The screenshot shows the Jenkins 'Getting Started' configuration page. It features five input fields: 'Username' (jay012), 'Password' (redacted), 'Confirm password' (redacted), 'Full name' (JAY RAVAL), and 'E-mail address' (jr1030442@gmail.com). The 'E-mail address' field is highlighted with a blue border. At the bottom, there are two buttons: 'Skip and continue as admin' and a larger 'Save and Continue' button.

Username  
jay012

Password  
.....

Confirm password  
.....

Full name  
JAY RAVAL

E-mail address  
jr1030442@gmail.com

Jenkins 2.528.3

Skip and continue as admin

Save and Continue

- This screen lets me create the first admin account to manage Jenkins securely.
- I've entered a custom username, full name, and email to personalize my Jenkins environment.
- The password I set will be used for future logins to the Jenkins dashboard.
- Clicking "Save and Continue" finalizes my admin setup and moves me to instance configuration.

# Instance Configuration

Jenkins URL:

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.

Jenkins 2.528.3

Not now

Save and Finish

- This screen sets the base URL Jenkins uses for links, notifications, and environment variables like .
- The default value is my EC2 public IP with port 8080, which works fine for direct access.
- If I later use a domain or reverse proxy, I'll want to update this URL accordingly.
- Clicking "Save and Finish" finalizes the setup and activates my Jenkins dashboard.

The screenshot shows the Jenkins 'Manage Jenkins' section, specifically the 'Plugins' page. A search bar at the top contains the text 'NodeJS plu'. On the left, a sidebar has tabs for 'Updates', 'Available plugins' (which is selected and highlighted in blue), 'Installed plugins', and 'Advanced settings'. The main area displays a table with columns: 'Install', 'Name ↓', 'Released', and 'Health'. One row is visible for the 'NodeJS 1.6.6' plugin, which is listed under the 'npm' category. The plugin was released '1 mo 29 days ago' and has a 'Health' score of '100' (indicated by a green circle). The 'Install' button is highlighted in blue.

- I selected the NodeJS plugin, which enables Jenkins to run Node.js scripts during builds.
- This is essential for automating tasks like testing, packaging, or deploying JavaScript-based applications.
- The plugin is actively maintained and reliable for my CI/CD pipelines.
- Once installed, I can configure NodeJS versions and use them in my job build steps.

The screenshot shows the Jenkins 'Plugins' page. On the left, there's a sidebar with links: 'Updates', 'Available plugins' (which is selected and highlighted in grey), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main area has a search bar at the top with the placeholder 'GitHub in' and a button with a magnifying glass icon. Below the search bar is a table header with columns: 'Install', 'Name ↓', 'Released', and 'Health'. A single row is shown in the table: 'GitHub Integration 0.7.3' by 'emailext' and 'Build Triggers', released '1 mo 3 days ago'. To the right of the release date is a green circular badge with the number '87'. At the top right of the main area are three buttons: 'Install' (with a download icon), a dropdown arrow, and a refresh icon.

Install	Name ↓	Released	Health
<input checked="" type="checkbox"/>	GitHub Integration 0.7.3 emailext Build Triggers GitHub Integration Plugin for Jenkins	1 mo 3 days ago	<span>87</span>

- I selected the GitHub Integration plugin, which enables Jenkins to interact with my GitHub repositories.
- It supports webhook triggers, pull request status updates, and Git-based build automation.
- The plugin integrates well with my CI/CD workflows.
- Once installed, I can link Jenkins jobs to GitHub projects for seamless code deployment.

The screenshot shows the Jenkins 'Manage Jenkins' interface with the 'Plugins' section selected. On the left, there's a sidebar with options: 'Updates', 'Available plugins' (which is selected and highlighted in grey), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main area has a search bar at the top with the placeholder 'Pipeline stage'. Below it is a table with columns: 'Install', 'Name ↓', 'Released', and 'Health'. A single row is shown for the 'Pipeline: Stage View 2.39' plugin, which is described as a 'User Interface' for the 'Pipeline Stage View Plugin'. The 'Install' button is visible at the top right of the table. The 'Released' timestamp is '1 day 9 hr ago' and the 'Health' status is '100' (indicated by a green circle).

- I selected the Pipeline: Stage View plugin, which adds a visual breakdown of pipeline stages in my Jenkins jobs.
- It helps me monitor each stage's status, duration, and logs in a clean UI.
- This is especially useful for debugging and optimizing my multi-step build processes.
- With this plugin, Jenkins becomes more intuitive for tracking my complex workflows.

The screenshot shows the Jenkins management interface for tools. It includes sections for Ant installations, Maven installations, and NodeJS installations, each with an 'Add' button. At the bottom are 'Save' and 'Apply' buttons.

Ant installations

+ Add Ant

Maven installations

+ Add Maven

NodeJS installations

+ Add NodeJS

Save    Apply

- Editing Node.js here lets me specify the name (used in jobs), the Node.js version, and optionally the installation path.
- This ensures Jenkins jobs can automatically use the correct Node.js runtime without needing manual setup on the EC2 instance.

Not secure 54.226.136.58:8080/manage/configureTools/

Jenkins / Manage Jenkins / Tools

Install automatically ?

**Install from nodejs.org**

Version

NodeJS 20.19.6

For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail

Force 32bit architecture

Global npm packages to install

Specify list of packages to install globally -- see `npm install -g`. Note that you can fix the packages version by using the syntax '`packageName@version`'

Global npm packages refresh hours

Duration, in hours, before 2 npm cache update. Note that 0 will always update npm cache

72

+ Add Installer

Save Apply

The screenshot shows the Jenkins 'Tools' configuration page. Under the 'Install from nodejs.org' section, the 'Install automatically' checkbox is checked. The 'Version' dropdown is set to 'NodeJS 20.19.6'. The 'Force 32bit architecture' checkbox is unchecked. The 'Global npm packages to install' and 'Global npm packages refresh hours' fields are empty. There is a '+ Add Installer' button and 'Save' and 'Apply' buttons at the bottom.

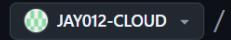
- I enabled automatic installation so Jenkins fetches Node.js directly from nodejs.org during job execution.
- I selected version 20.19.6, ensuring consistent runtime across all builds.

## Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).  
Required fields are marked with an asterisk (\*).

### 1 General

Owner \*



Repository name \*

fastfood-cicd-JAY

fastfood-cicd-JAY is available.

Great repository names are short and memorable. How about [fluffy-system](#)?

Description

0 / 350 characters

### 2 Configuration

Choose visibility \*

Choose who can see and commit to this repository

Public

Add README

READMEs can be used as longer descriptions. [About READMEs](#)

Off

Add .gitignore

.gitignore tells git which files not to track. [About ignoring files](#)

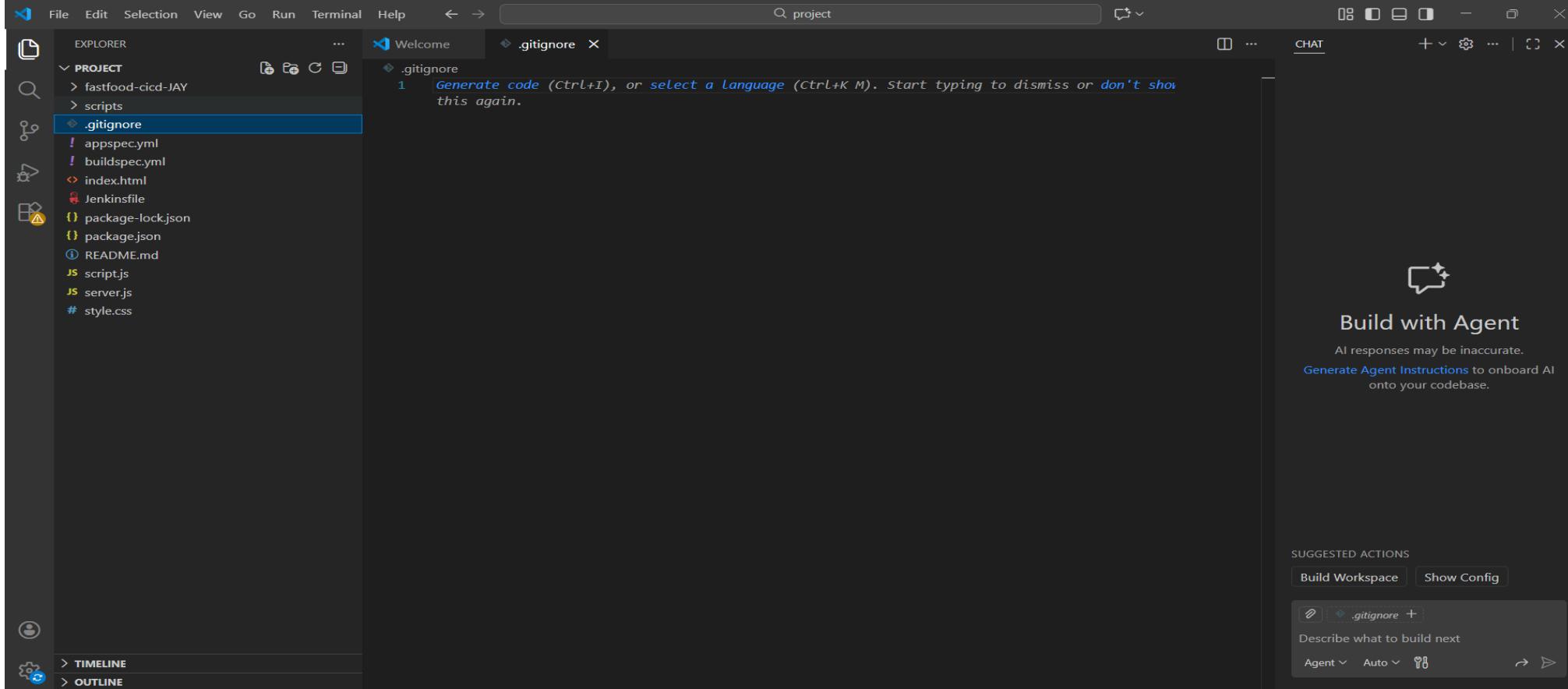
No .gitignore

- I created a repository named fastfood-cicd-JAY to store my application code and CI/CD configuration files.
- This repo will hold important files like Jenkinsfile, buildspec.yml, and appspec.yml, which define my automation pipeline.

```
ubuntu@ip-172-31-28-21:~$ mkdir project
ubuntu@ip-172-31-28-21:~$ cd project
ubuntu@ip-172-31-28-21:~/project$ git clone https://github.com/JAY012-CLOUD/fastfood-cicd-JAY
Cloning into 'fastfood-cicd-JAY'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
ubuntu@ip-172-31-28-21:~/project$ cd fastfood-cicd-JAY
ubuntu@ip-172-31-28-21:~/project/fastfood-cicd-JAY$
```

**cd downloads** I'm navigating to my downloads directory where I'll clone the FastFood CI/CD project repository from GitHub.

I'm cloning an existing FastFood CI/CD project repository from GitHub to my local machine. This downloads all project files, commit history, and branches to use as a starting template.



- I opened my project folder in VS Code, showing key files like Jenkinsfile, buildspec.yml, and appspec.yml for CI/CD automation.
- The presence of package.json and server.js confirms this is a Node.js-based web app.
- The Jenkinsfile will define my pipeline stages, while buildspec.yml and appspec.yml support AWS CodeBuild and CodeDeploy.
- This setup is ready to connect with GitHub and Jenkins for automated testing and deployment.

```
<html lang="en">
  <body>
    <div class="container">
      <div class="header">
        <div class="header-content">
          <div class="header-section">
            <h1>Quick Links</h1>
            <ul>
              <li><a href="#menu">Menu</a></li>
              <li><a href="#about">About</a></li>
              <li><a href="#contact">Contact</a></li>
            </ul>
          </div>
          <div class="header-section">
            <h2>Follow Us</h2>
            <div class="social-links">
              <a href="#"><i class="fab fa-facebook"></i></a>
              <a href="#"><i class="fab fa-instagram"></i></a>
              <a href="#"><i class="fab fa-twitter"></i></a>
            </div>
          </div>
        </div>
        <div class="header-bottom">
          <p>&copy; 2025 Quickbite. Built with Jenkins & AWS CI/CD Pipeline</p>
          <p>Build Version: <span id="buildVersion">1.0</span></p>
        </div>
      </div>
    </div>
    <script src="script.js"></script>
  </body>
</html>
```

Build with Agent  
AI responses may be inaccurate.  
Generate Agent Instructions to onboard AI onto your codebase.

SUGGESTED ACTIONS  
Build Workspace Show Config

## index.html

This is the main HTML file for your web app. It includes navigation, content sections, and a footer that displays build metadata like version and CI/CD tools used (Jenkins & AWS).

## Jenkinsfile

Defines your CI/CD pipeline stages. Jenkins reads this file to know how to build, test, and deploy your app. It's the heart of your automation workflow.

## buildspec.yml

Used by AWS CodeBuild to define build commands, environment variables, and artifacts. It ensures your app is built consistently in the cloud.

## appspec.yml

Used by AWS CodeDeploy to manage deployment steps, hooks, and file locations. It tells AWS how to deploy your app to EC2 or other targets.

## package.json

Lists your Node.js dependencies, scripts, and metadata. It's essential for installing packages and running build/test commands.

## package-lock.json

Locks the exact versions of dependencies to ensure consistent builds across environments.

## server.js

Your Node.js backend file. It likely handles routing, server logic, and API endpoints for your web app.

## README.md

Provides documentation for your project — purpose, setup instructions, and usage notes. Useful for collaborators and future reference.

## script.js

Contains client-side JavaScript for interactivity, such as handling user actions or dynamic content updates.

## style.css

Defines the visual styling of your web app — layout, colors, fonts, and responsiveness.

## scripts/

A folder likely containing deployment or utility scripts used in your CI/CD pipeline or local development.

## .gitignore

Specifies which files and folders Git should ignore (e.g., logs, temp files). Keeps your repo clean and focused.

JAY012-CLOUD / fastfood-cicd--JAY

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

fastfood-cicd--JAY Public

main 1 Branch 0 Tags

Go to file Add file Code

**JAY012-CLOUD** fast food files added 736b1e0 · 20 hours ago 7 Commits

script	Delete script/temp	20 hours ago
.gitignore	fast food files added	20 hours ago
Jenkinsfile	fast food files added	20 hours ago
README.md	Create README.md	yesterday
appspec.yml	fast food files added	20 hours ago
buildspec.yml	fast food files added	20 hours ago
index.html	fast food files added	20 hours ago
package-lock.json	fast food files added	20 hours ago
package.json	fast food files added	20 hours ago
script.js	fast food files added	20 hours ago
server.js	fast food files added	20 hours ago

About

for project

- Readme
- Activity
- 0 stars
- 0 watching
- 0 forks

Releases

No releases published [Create a new release](#)

Packages

No packages published [Publish your first package](#)

Languages

CSS 39.0% JavaScript 37.6%

- I pushed my CI/CD project to GitHub under the repo name **fastfood-cicd--JAY**, hosted publicly on the main branch.
- The repo contains key files like **Jenkinsfile**, **buildspec.yml**, **appspec.yml**, and **server.js** — all essential for automated deployment.
- With 7 commits so far, I've begun versioning my code and tracking changes, including cleanup and initial setup.
- This repo is now ready to integrate with Jenkins for continuous build and deployment from GitHub to AWS.

```
ubuntu@ip-172-31-30-214: ~      +  -  ×
GNU nano 7.2                               /etc/sudoers.tmp *
```

```
# "sudo scp" or "sudo rsync" should be able to use your SSH agent.
#Defaults:%sudo env_keep += "SSH_AGENT_PID SSH_AUTH_SOCK"

# Ditto for GPG agent
#Defaults:%sudo env_keep += "GPG_AGENT_INFO"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin  ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "@include" directives:

@include /etc/sudoers.d
jenkins ALL=(ALL) NOPASSWD: ALL

^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute        ^C Location       M-U Undo        M-A Set Mark
^X Exit          ^R Read File     ^\ Replace       ^U Paste          ^J Justify        ^/ Go To Line    M-E Redo        M-6 Copy
```

- I edited the sudoers file to grant the jenkins user passwordless sudo access.
- The line added is: jenkins ALL=(ALL) NOPASSWD: ALL
- This allows Jenkins to run commands with elevated privileges during automation tasks without asking for a password.
- It is useful for CI/CD pipelines that need system-level operations like deployments, file transfers, or service restarts.
- I made sure to edit it safely using visudo to avoid syntax errors that could break sudo access.

```
ubuntu@ip-172-31-30-214: ~  + | - X
Jan 15 09:03:02 ip-172-31-30-214 systemd[1]: Started codedeploy-agent.service - LSB: AWS CodeDeploy Host Agent.
ubuntu@ip-172-31-30-214:/tmp$ client_loop: send disconnect: Connection reset
PS C:\Users\JAY\downloads> ssh -i "Last-004-rem.pem" ubuntu@ec2-54-226-140-111.compute-1.amazonaws.com
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of Thu Jan 15 09:10:29 UTC 2026

System load:  0.04          Temperature:      -273.1 C
Usage of /:   18.7% of 18.33GB  Processes:        116
Memory usage: 81%           Users logged in:   0
Swap usage:   0%            IPv4 address for ens5: 172.31.30.214

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

*** System restart required ***
Last login: Thu Jan 15 08:53:09 2026 from 49.36.83.23
ubuntu@ip-172-31-30-214:~$ sudo usermod -aG ubuntu jenkins
ubuntu@ip-172-31-30-214:~$ sudo visudo
ubuntu@ip-172-31-30-214:~$ |
```

**sudo usermod -aG ubuntu jenkins** I'm adding the jenkins user to the ubuntu group, granting Jenkins permission to access files owned by the ubuntu user. This is necessary for Jenkins to deploy and manage application files.

**sudo visudo** I'm opening the sudoers configuration file using the safe visudo editor to grant Jenkins sudo permissions if needed. This allows Jenkins to execute privileged commands during deployment.

```
ubuntu@ip-172-31-30-214:~$ sudo systemctl restart jenkins
```

```
ubuntu@ip-172-31-30-214:~$ |
```

**sudo systemctl restart jenkins** I'm restarting the Jenkins service to apply the user group membership changes made in the previous step. Jenkins needs to restart to recognize its new group permissions.

The screenshot shows the Jenkins main dashboard at the URL 54.226.140.111:8080. The page title is "Not secure". On the left sidebar, there are links for "New Item" and "Build History". Below the sidebar, there are two sections: "Build Queue" (which shows "No builds in the queue") and "Build Executor Status" (which shows "0/2"). The main content area features a "Welcome to Jenkins!" header, a message about displaying Jenkins jobs, and a "Start building your software project" section. This section includes a "Create a job" button, a "Set up a distributed build" section with "Set up an agent" and "Configure a cloud" options, and a link to "Learn more about distributed builds".

REST API Jenkins 2.528.3

- I accessed Jenkins via browser at IP 54.226.140.111:8080 and reached the main dashboard.
- The interface confirms Jenkins is running and ready to manage CI/CD jobs.



## New Item

Enter an item name

Select an item type

**Freestyle project**

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

**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.

Cancel | Help | Back | Forward | New Item

OK

- I created a new Jenkins job named WEB-CI-CD-PIPELINE to manage the CI/CD flow for my web application.
- I selected the Pipeline type, which is ideal for defining multi-stage automation like build, test, and deploy.

The screenshot shows the Jenkins configuration interface for the 'WEB-CI-CD-PIPELINE' job. The left sidebar lists 'Configure' sections: General, Triggers (selected), Pipeline, and Advanced. The main content area has a header: 'Set up automated actions that start your build based on specific events, like code changes or scheduled times.' Below this is a list of triggers with 'GitHub hook trigger for GITScm polling' checked. A 'Pipeline' section follows, with a sub-header 'Definition' showing 'Pipeline script from SCM' selected. Under 'SCM', 'None' is chosen. At the bottom are 'Save' and 'Apply' buttons.

- I configured the WEB-CI-CD-PIPELINE job to trigger automatically using the GitHub hook trigger for GITScm polling.
- This means Jenkins will start a build whenever changes are pushed to my GitHub repository.
- I selected “Pipeline script from SCM” to pull the Jenkinsfile directly from source control.
- The SCM option is currently set to “None” — I will update this to Git and provide the repository URL and credentials.
- Once configured, Jenkins will fetch the pipeline definition and execute the CI/CD stages on every code update.

The screenshot shows the GitHub settings interface for a repository named 'fastfood-cicd--JAY'. The left sidebar is collapsed, and the main navigation bar includes 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings' (which is currently selected). The left-hand sidebar under 'Webhooks' is expanded, showing options like 'General', 'Access', 'Collaborators', 'Moderation options', 'Code and automation' (with 'Branches', 'Tags', 'Rules', 'Actions', 'Models', and 'Copilot'), 'Environments', 'Codespaces', 'Pages', 'Security', and 'Advanced Security'. The 'Webhooks' section is highlighted with a blue border. The right panel displays the 'Webhooks / Add webhook' configuration form. It contains a descriptive text about sending POST requests to a URL with event details, a link to developer documentation, and fields for 'Payload URL' (set to 'http://54.226.140.111:8080/job/github-webhook/'), 'Content type' (set to 'application/json'), and a 'Secret' field. Below these are sections for 'SSL verification' (with 'Enable SSL verification' selected) and 'Which events would you like to trigger this webhook?' (with 'Just the push event.' selected).

- I added a webhook in my GitHub repository fastfood-cicd--JAY to integrate with Jenkins.

[Status](#)[Changes](#)[Build Now](#)[Configure](#)[Delete Pipeline](#)[Full Stage View](#)[Stages](#)[Rename](#)[Pipeline Syntax](#)[GitHub Hook Log](#)

## WEB-CI-CD-PIPELINE

My web ci/cd pipeline

[Edit description](#)

### Stage View



### Permalinks

Builds

Today

① #1 9:25 AM

[REST API](#)

Jenkins 2.528.3

- After build
- I viewed the stage execution summary for the WEB-CI-CD-PIPELINE job in Jenkins.

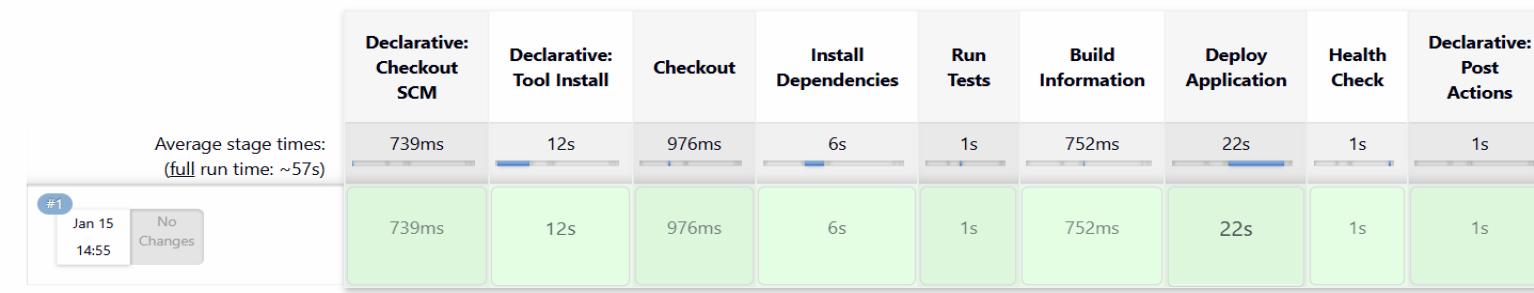
[Status](#)[Changes](#)[Build Now](#)[Configure](#)[Delete Pipeline](#)[Full Stage View](#)[Stages](#)[Rename](#)[Pipeline Syntax](#)[GitHub Hook Log](#)

## WEB-CI-CD-PIPELINE

My web ci/cd pipeline

[Edit description](#)

### Stage View



Builds ...

Today

#1 9:25 AM

### Permalinks

[REST API](#) Jenkins 2.528.3

- Build #1 executed today at 9:25 AM completed successfully.
- This confirms that all CI/CD stages are functioning and the deployment pipeline is stable.

```
ubuntu@ip-172-31-30-214: ~  + | 
Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

*** System restart required ***
Last login: Thu Jan 15 09:10:31 2026 from 49.36.83.23
ubuntu@ip-172-31-30-214:~$ which pm2
/usr/bin/pm2
ubuntu@ip-172-31-30-214:~$ ls -la /var/www/fastfood-app/
total 128
drwxr-xr-x  4 jenkins jenkins  4096 Jan 15 09:25 .
drwxr-xr-x  3 root   root    4096 Jan 15 09:25 ..
-rw-r--r--  1 jenkins jenkins 10902 Jan 15 09:25 Jenkinsfile
-rw-r--r--  1 jenkins jenkins    47 Jan 15 09:25 README.md
-rw-r--r--  1 jenkins jenkins   732 Jan 15 09:25 appspec.yml
-rw-r--r--  1 jenkins jenkins 14115 Jan 15 09:25 buildspec.yml
-rw-r--r--  1 jenkins jenkins 10814 Jan 15 09:25 index.html
drwxr-xr-x 70 jenkins jenkins 4096 Jan 15 09:25 node_modules
-rw-r--r--  1 jenkins jenkins 29362 Jan 15 09:25 package-lock.json
-rw-r--r--  1 jenkins jenkins   437 Jan 15 09:25 package.json
-rw-r--r--  1 jenkins jenkins 11860 Jan 15 09:25 script.js
drwxr-xr-x  2 jenkins jenkins  4096 Jan 15 09:25 scripts
-rw-r--r--  1 jenkins jenkins  5348 Jan 15 09:25 server.js
-rw-r--r--  1 jenkins jenkins 17494 Jan 15 09:25 style.css
ubuntu@ip-172-31-30-214:~$ |
```

**ls -la /var/www/ | grep fastfood-app** I'm listing the /var/www/ directory with detailed information and filtering for fastfood-app to verify the directory exists and check its permissions and ownership.



```
[Pipeline] {
[Pipeline] sh
+ echo ⚒ Final PM2 Status:
⚡ Final PM2 Status:
+ pm2 list
| id | name | namespace | version | mode | pid | uptime | ⚡ | status | cpu | mem | user | watching |
| ⚡[1m@[36m@[39m@[22m | fastfood-app | default | 1.0.0 | ⚡[7m@[1mfork@[22m@[27m | 18435 | 10s | ⚡ | disabled@[39m | 0 |
| ⚡[32m@[1monline@[22m@[39m | 0% | 25.9mb | ⚡[1mjenkins@[22m | ⚡[90mdisabled@[39m |
| ⚡[1m
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

- The pipeline completed successfully, confirming that the application is running and managed correctly by PM2.

Not secure 54.226.140.111:8080/job/WEB-CI-CD-PIPELINE/1/console

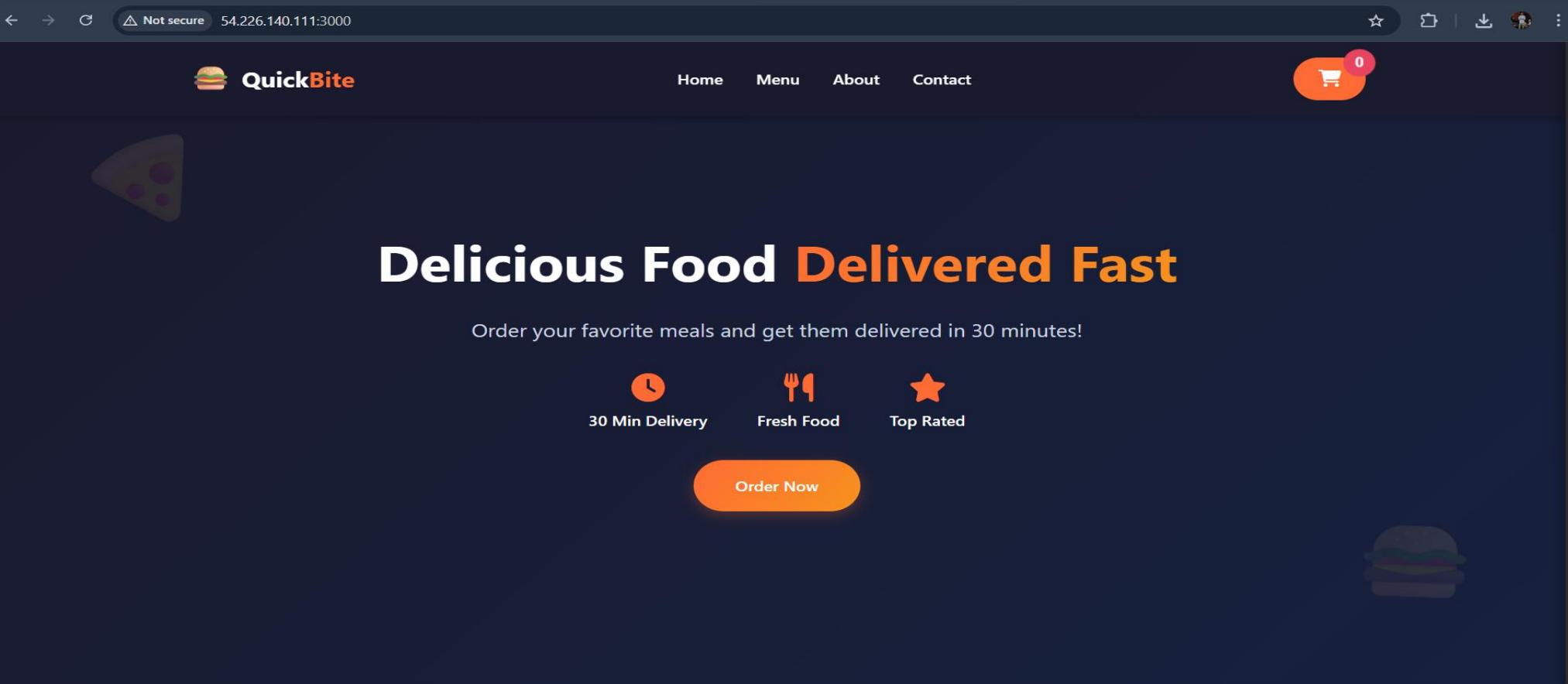
Jenkins / WEB-CI-CD-PIPELINE #1 / Console Output

```
[Pipeline] sh
+ curl -s ifconfig.me
[Pipeline] echo
Application URL: http://54.226.140.111:3000
[Pipeline] echo
Health Check: http://54.226.140.111:3000/health
[Pipeline] echo
API Menu: http://54.226.140.111:3000/api/menu
[Pipeline] }
[Pipeline] // script
[Pipeline] echo
=====
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ echo 🎉 Final PM2 Status:
🎉 Final PM2 Status:
+ pm2 list

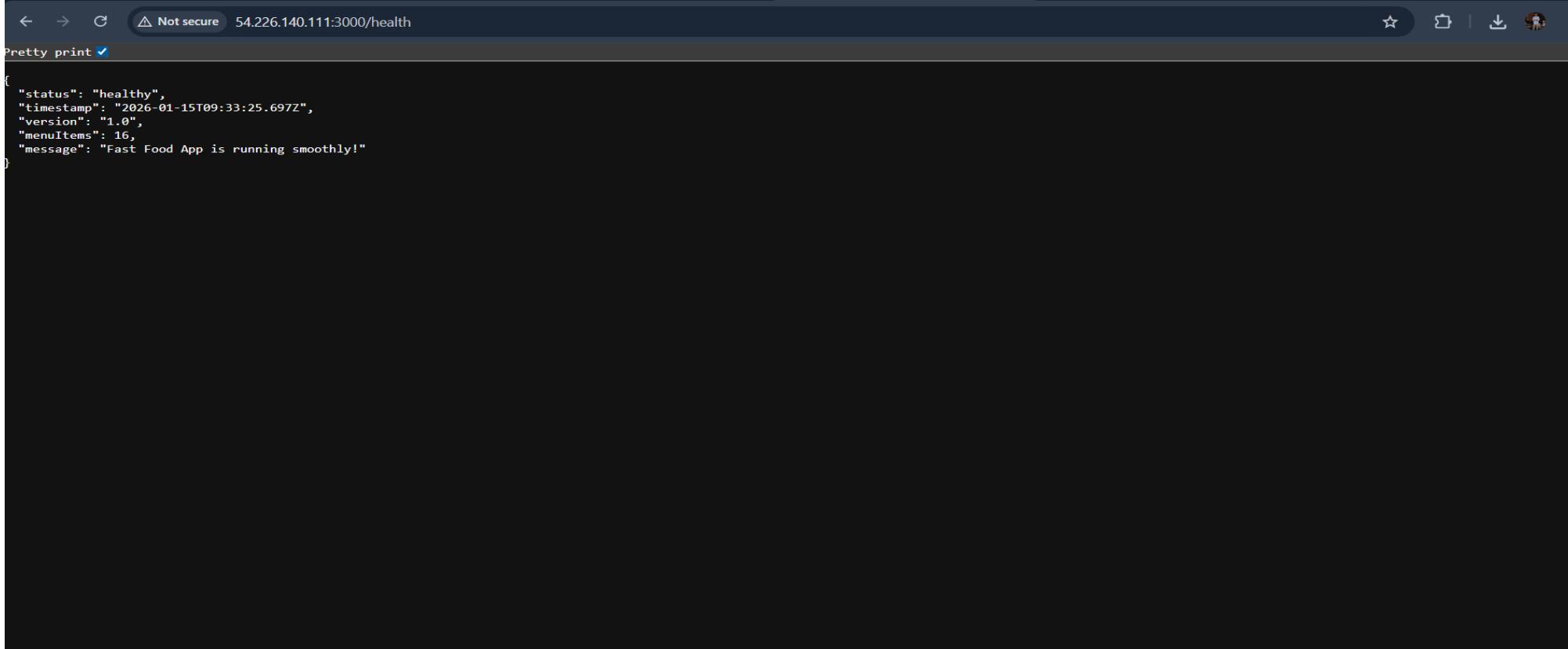

| id                | name         | namespace | version | mode         | pid            | uptime | σ   | status | cpu | mem | user | watching |
|-------------------|--------------|-----------|---------|--------------|----------------|--------|-----|--------|-----|-----|------|----------|
| 1m039m22m         | fastfood-app |           | default | 1.0.0        | 7m1mfork22m27m | 18435  | 10s | 0      |     |     |      |          |
| 32m1monline22m39m |              | 0%        | 25.9mb  | 1mjenkins22m | 90mdisabled39m |        |     |        |     |     |      |          |


[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
```

- I used to fetch the public IP address of the deployed EC2 instance.
- Jenkins printed the following deployment endpoints:
  - Application URL: (54.226.140.111 in Bing)
  - Health Check: (54.226.140.111 in Bing)
  - API Menu: (54.226.140.111 in Bing)
- These URLs confirm that the Node.js application is live and accessible over the internet.



- I accessed the deployed web application at 54.226.140.111:3000 and confirmed the homepage is live.
- This confirms the CI/CD pipeline successfully deployed the Node.js application and it is publicly accessible.

A screenshot of a web browser window. The address bar shows the URL "54.226.140.111:3000/health". The page content is a JSON object with the following structure:

```
{  "status": "healthy",  "timestamp": "2026-01-15T09:33:25.697Z",  "version": "1.0",  "menuItems": 16,  "message": "Fast Food App is running smoothly!"}
```

The "Pretty print" checkbox is checked, which is indicated by a checked box icon next to the label.

- This confirms the deployment is successful and the backend is responding correctly to health probes.

```
 Pretty print  [Pretty print  ]  
[  
 {  
   "id": 1,  
   "name": "Classic Burger",  
   "category": "burgers",  
   "price": 299,  
   "description": "Juicy aloo patty with lettuce, tomato, and special sauce",  
   "image": "🍔"  
 },  
 {  
   "id": 2,  
   "name": "Cheese Burger",  
   "category": "burgers",  
   "price": 349,  
   "description": "Double cheese with premium aloo patty",  
   "image": "🧀"  
 },  
 {  
   "id": 3,  
   "name": "Chicken Burger",  
   "category": "burgers",  
   "price": 279,  
   "description": "Crispy chicken fillet with mayo and lettuce",  
   "image": "🍗"  
 },  
 {  
   "id": 4,  
   "name": "Veggie Burger",  
   "category": "burgers",  
   "price": 249,  
   "description": "Plant-based patty with fresh vegetables",  
   "image": "🥕"  
 },  
 {  
   "id": 5,  
   "name": "Margherita Pizza",  
   "category": "pizza",  
   "price": 399,  
   "description": "Classic cheese pizza with fresh basil",  
   "image": "🍕"  
 },  
 {  
   "id": 6,  
   "name": "Pepperoni Pizza",  
   "category": "pizza",  
   "price": 449,  
   "description": "Pepperoni and cheese pizza",  
   "image": "🍕"  
 }  
 ]
```

- This confirms the backend API is functioning and delivering structured menu data for frontend rendering and order processing.



The screenshot shows the AWS Console Home page. In the top left, there's a 'Recently visited' section with icons for various services: EC2, VPC, Support, Billing and Cost Management, CodeBuild, CodeDeploy, IAM (which has a handwritten note '① click on IAM' pointing to it), and CodePipeline. To the right is an 'Applications' section showing zero applications, with a 'Create application' button. Below these are sections for 'Welcome to AWS' (with links to CloudShell, Feedback, and Console Mobile App) and 'AWS Health' (showing open issues). At the bottom, there's a footer with links for Current month, Cost (\$), Privacy, Terms, and Cookie preferences.

- IAM is highlighted as the first step, since roles and permissions must be configured before setting up CI/CD.

- Step 1  
Select trusted entity
- Step 2  
**Add permissions**
- Step 3  
Name, review, and create

## Add permissions Info

### Permissions policies (1/1141) Info

Choose one or more policies to attach to your new role.

Filter by Type

AmazonS3ReadOnlyAccess X

All types

1 match

Policy name L2

 [AmazonS3ReadOnlyAccess](#)

Type

AWS managed

Description

Provides read only access to all buckets v...

► Set permissions boundary - optional

Cancel

Previous

Next

Select this policy

- I selected the AWS managed policy named AmazonS3ReadOnlyAccess.
- This policy provides read-only access to all S3 buckets in the account.
- It's useful for services like CodeBuild that need to fetch build artifacts or configuration files from S3 without modifying them.

Step 1  
Select trusted entity

Step 2  
**Add permissions**

Step 3  
Name, review, and create

## Add permissions Info

Permissions policies (2/1141) Info

Choose one or more policies to attach to your new role.

Filter by Type

CloudWatchLogsFullAccess  All types 1 match

Policy name ▾ Type Description

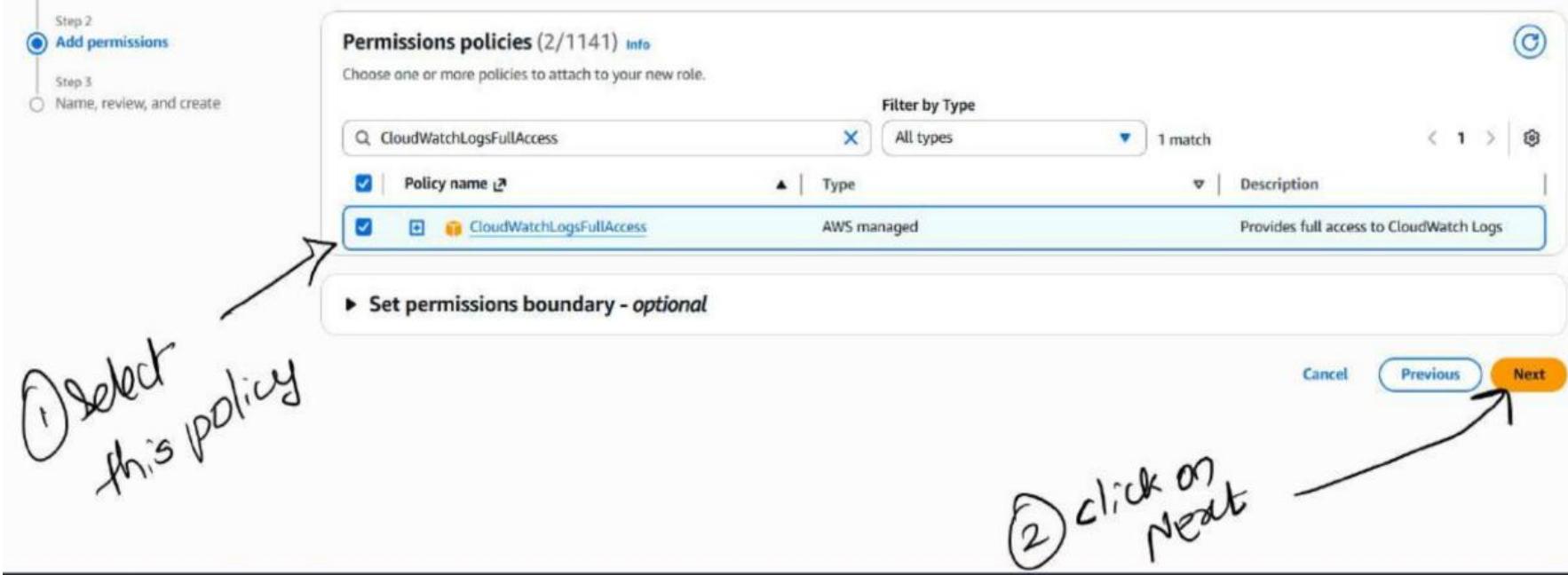
CloudWatchLogsFullAccess AWS managed Provides full access to CloudWatch Logs

▶ Set permissions boundary - optional

① Select this policy

② click on next

Cancel Previous **Next**



- I selected the AWS managed policy named CloudWatchLogsFullAccess.
- This policy provides full access to CloudWatch Logs, allowing services like CodeBuild to write detailed logs during build execution.
- It's essential for monitoring build output and debugging failures.

Role EC2-CodeDeploy-Role created.

View role

- Step 1  
Select trusted entity
- Step 2  
**Add permissions**
- Step 3  
Name, review, and create

## Add permissions Info

### Permissions policies (1) Info

The type of role that you selected requires the following policy.

Policy name

AWSCodeDeployRole

Type

AWS managed

► Set permissions boundary - optional

(2) Click on next

Cancel

Previous

Next

This policy will be automatically selected

- The AWS managed policy AWSCodeDeployRole was automatically selected.
- This policy grants EC2 permission to work with CodeDeploy during deployments.

Step 1  
Select trusted entity

Step 2  
**Add permissions**

Step 3  
Name, review, and create

## Add permissions Info

Permissions policies (1/1141) Info

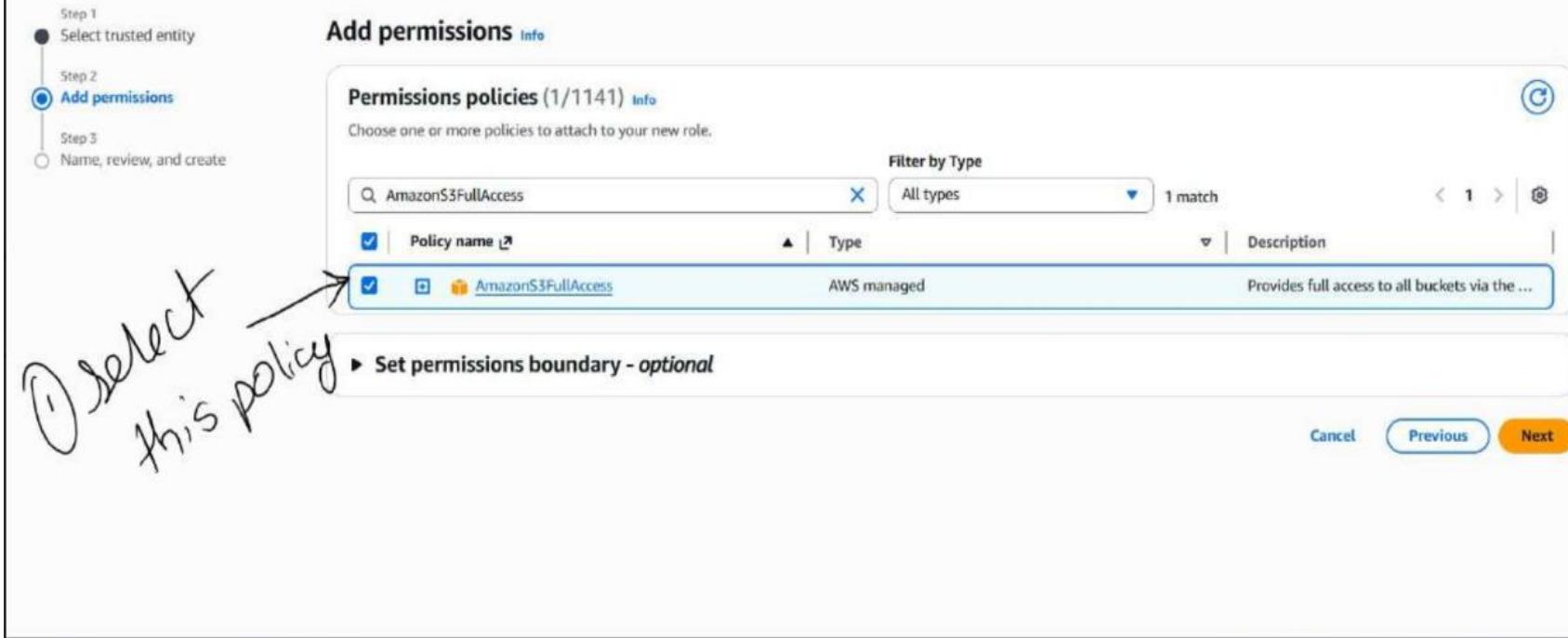
Choose one or more policies to attach to your new role.

Filter by Type

Policy name	Type	Description
<input checked="" type="checkbox"/> AmazonS3FullAccess	AWS managed	Provides full access to all buckets via the ...

① Select this policy → Set permissions boundary - optional

Cancel Previous Next



- I selected the AWS managed policy named AmazonS3FullAccess.
- This policy provides full access to all S3 buckets in the account.
- It's essential for services like CodeBuild and CodeDeploy that need to read and write build artifacts or deployment packages.
- This permission allows uploading, downloading, and managing files in S3 during CI/CD execution.

Step 1  
Select trusted entity

Step 2  
**Add permissions**

Step 3  
Name, review, and create

### Add permissions Info

Permissions policies (2/1141) Info

Choose one or more policies to attach to your new role.

Filter by Type

CloudWatchLogsFullAccess  All types 1 match

Policy name ▾ Type Description

CloudWatchLogsFullAccess AWS managed Provides full access to CloudWatch Logs

Select this policy

② Click on Next

Set permissions boundary - optional

Cancel Previous Next

- I selected the AWS managed policy named CloudWatchLogsFullAccess.
- This policy provides full access to CloudWatch Logs, allowing services like CodeBuild to write logs during build execution.
- It's critical for monitoring build output and troubleshooting errors in the CI/CD pipeline.

IAM > Roles > CodeBuild-Service-Role

Identity and Access Management (IAM)

CodeBuild-Service-Role [Info](#)

Allows CodeBuild to call AWS services on your behalf.

Search IAM

Dashboard

Access management

- User groups
- Users
- Roles**
- Policies
- Identity providers
- Account settings
- Root access management
- Temporary delegation requests
- New

Access reports

- Access Analyzer
- Resource analysis [New](#)
- Unused access

Summary

Creation date: January 02, 2026, 17:03 (UTC+05:30)

Last activity: -

ARN: arn:aws:iam::586549103866:role/CodeBuild-Service-Role

Maximum session duration: 1 hour

Edit

Permissions [Edit](#) Trust relationships Tags Last Accessed Revoke sessions

Permissions policies (2) [Info](#)

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AmazonS3FullAccess	AWS managed	1
CloudWatchLogsFullAccess	AWS managed	2

Simulate Remove Add permissions ▾ Attach policies Create inline policy

CloudShell Feedback Console Mobile App © 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

① Select this option

- I'm now ready to add a custom inline policy , using the “Create inline policy” option under Add permissions.

Step 1  
 **Specify permissions**  
 Step 2  
 Review and create

**Specify permissions** Info

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

**Policy editor**

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "s3:*",
8         "logs:*,"
9         "codebuild:*,"
10      ],
11      "Resource": "*"
12    }
13  ]
14 }
```

**Select JSON**Visual  **JSON** Actions ▾**Edit statement****Select a statement**

Select an existing statement in the policy or add a new statement.

**+ Add new statement***② Add this policy**③ Scroll down and click on next*

- I created a custom inline policy named CodeBuild-Policy.
- The policy grants full access to three services:
- S3 → for storing and retrieving build artifacts.
- CloudWatch Logs → for logging build output.
- CodeBuild → for managing build projects and execution.
- All permissions apply to all resources in the account.

Step 2  
Review and create**Policy details****Policy name**

Enter a meaningful name to identify this policy.

CodeBuild-Policy

Maximum 128 characters. Use alphanumeric and '+-\_@.' characters.

① Give the policy name

**Permissions defined in this policy** Info**Edit**

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Q Search

**Allow (3 of 461 services)** Show remaining 458 services

Service	Access level	Resource	Request condition
CloudWatch Logs	Full access	All resources	None
CodeBuild	Full access	All resources	None
S3	Full access	All resources	None

**Create policy**

③ Click on create policy

- All permissions apply to all resources in the account.
- No request conditions were defined, keeping the policy broad and flexible.
- I clicked “Create policy” to finalize and attach it to the CodeBuild-Service-Role.

```
ubuntu@ip-172-31-3-81:/tmp$ sudo systemctl start codedeploy-agent
ubuntu@ip-172-31-3-81:/tmp$ sudo systemctl enable codedeploy-agent
codedeploy-agent.service is not a native service, redirecting to systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable codedeploy-agent
ubuntu@ip-172-31-3-81:/tmp$ sudo systemctl status codedeploy-agent
● codedeploy-agent.service - LSB: AWS CodeDeploy Host Agent
    Loaded: loaded (/etc/init.d/codedeploy-agent; generated)
    Active: active (running) since Fri 2026-01-02 11:00:04 UTC; 1h 48min ago
      Docs: man:systemd-sysv-generator(8)
     Tasks: 3 (limit: 1017)
   Memory: 69.4M (peak: 82.4M)
     CPU: 2.361s
    CGroup: /system.slice/codedeploy-agent.service
            └─937 "codedeploy-agent: master 937"
                ├─940 "codedeploy-agent: InstanceAgent::Plugins::CodeDeployPlugin::CommandPoller of master 937"
Jan 02 11:00:00 ip-172-31-3-81 systemd[1]: Starting codedeploy-agent.service - LSB: AWS CodeDeploy Host Agent...
Jan 02 11:00:04 ip-172-31-3-81 codedeploy-agent[531]: Starting codedeploy-agent:
Jan 02 11:00:04 ip-172-31-3-81 systemd[1]: Started codedeploy-agent.service - LSB: AWS CodeDeploy Host Agent.
ubuntu@ip-172-31-3-81:/tmp$
```

① Start the code deploy-agent  
② check the status

- I enabled the agent to run automatically on boot using `systemctl enable`.
- Checking the status confirmed the agent is active and running, with processes for the master and the command poller.

```
ubuntu@ip-172-31-3-81:~$ sudo chown -R ubuntu:ubuntu /var/www/fastfood-app ←① it should be owned by  
ubuntu  
ubuntu@ip-172-31-3-81:~$ ls -la /var/www/ | grep fastfood-app  
drwxr-xr-x 4 ubuntu ubuntu 4096 Jan  2 11:03 fastfood-app ←② Verify the ownership ubuntu  
drwxr-xr-x 4 root   root   4096 Jan  1 18:32 fastfood-app.backup.20260101_183227  
drwxr-xr-x 4 root   root   4096 Jan  1 19:20 fastfood-app.backup.20260101_192024  
drwxr-xr-x 4 root   root   4096 Jan  1 19:23 fastfood-app.backup.20260101_192321  
drwxr-xr-x 4 root   root   4096 Jan  1 19:32 fastfood-app.backup.20260101_193258  
drwxr-xr-x 4 root   root   4096 Jan  1 19:36 fastfood-app.backup.20260101_193658  
drwxr-xr-x 4 root   root   4096 Jan  1 19:42 fastfood-app.backup.20260101_194218  
drwxr-xr-x 4 root   root   4096 Jan  1 19:46 fastfood-app.backup.20260101_194623  
drwxr-xr-x 4 root   root   4096 Jan  1 19:49 fastfood-app.backup.20260101_194943  
drwxr-xr-x 4 root   root   4096 Jan  1 19:56 fastfood-app.backup.20260101_195623  
drwxr-xr-x 4 root   root   4096 Jan  1 20:06 fastfood-app.backup.20260101_200643  
drwxr-xr-x 4 root   root   4096 Jan  1 20:24 fastfood-app.backup.20260101_202438  
drwxr-xr-x 4 root   root   4096 Jan  1 20:55 fastfood-app.backup.20260101_205530  
drwxr-xr-x 4 root   root   4096 Jan  1 21:09 fastfood-app.backup.20260101_210956  
drwxr-xr-x 4 root   root   4096 Jan  2 11:03 fastfood-app.backup.20260102_110345
```

- I changed ownership of the /var/www/fastfood-app directory to ubuntu:ubuntu using chown.
- This ensures the ubuntu user has full control over the live application directory.
- I verified with ls -la and confirmed fastfood-app is owned by ubuntu.

Storage

# Amazon S3

## Store and retrieve any amount of data from anywhere

Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance.

**Create a bucket**

Every object in S3 is stored in a bucket. To upload files and folders to S3, you'll need to create a bucket where the objects will be stored.

**Create bucket**

**Pricing**

With S3, there are no minimum fees. You only pay for what you use. Prices are based on the location of your S3 bucket.

Estimate your monthly bill using the [AWS Simple Monthly Calculator ↗](#)

[View pricing details ↗](#)

**How it works**



- S3 offers scalable object storage with high availability, security, and performance.
- To begin, I clicked on Create bucket

## Create bucket Info

Buckets are containers for data stored in S3.

### General configuration

#### AWS Region

US East (N. Virginia) us-east-1

#### Bucket type Info

##### General purpose

Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Give the bucket name scroll down and create bucket

##### Directory

Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

#### Bucket name Info

fastfood-cicd-artifacts-vighnesh

Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Learn more](#)

#### Copy settings from existing bucket - optional

Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Format: s3://bucket/prefix

### Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

- Bucket type: General purpose, suitable for storing build and deployment artifacts.
- Bucket name entered.

## Create build project

① Give the project name

Project configuration

Project name

FastFood-Build

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

Project type

Select what type of project you would like to create. [Info](#)

Default project  
Create a custom CodeBuild project.

Runner project  
Create a CodeBuild managed runner for workflows in GitHub Actions, GitHub Enterprise Actions, GitLab, or Buildkite.

② Give the description and scroll down

▼ Additional configuration

Description, public build access, build badge, concurrent build limit, tags

Description - optional

Build and test FastFood application

Public build access - optional

Public build access allows you to make the build results, including logs and artifacts, for this project available for the general public.

Enable public build access

Build badge - optional

Enable build badges

CloudShell Feedback Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

- I opened the AWS CodeBuild console to create a new build project.
- I entered the project name as FastFood-Build.
- This name uniquely identifies the build process for the FastFood application.
- In the description field, I wrote “Build and test FastFood application” to clarify the project’s purpose.

Add tag Remove tag

Add source

Source ▾

Source 1 - Primary

Source provider

GitHub

Credential

You have not connected to GitHub. Manage account credentials.

Use override credentials for this project only

Repository

- I selected GitHub as the source provider for the build project.
- This allows CodeBuild to pull source code directly from a GitHub repository.
- Since the account was not yet connected, a warning appeared stating that GitHub credentials were missing.
- I clicked on “Manage account credentials” to begin the GitHub authorization process.

## Manage default source credential

### Source Provider

GitHub

### Credential type

GitHub App

Connect project to GitHub using an AWS managed GitHub App

Personal access token

Connect project to GitHub using a personal access token

OAuth app

Connect project to GitHub using an OAuth app

### Connection

① Select GitHub APP

- I selected GitHub as the source provider.
- Under credential type, I chose “GitHub App” to connect using AWS’s managed GitHub integration.
- This method simplifies authentication and allows CodeBuild to securely access the repository.
- It avoids manual token management and supports automated builds triggered by GitHub events.
- This step completes the GitHub connection setup for the build project.

Repository

Q https://github.com/vighnesh32/fastfood-cicd-project ✖

Source version - optional Info

Enter a pull request, branch, commit ID, tag, or reference and a commit ID.

② select your repo and scroll down

▼ Additional configuration

Git clone depth, Git submodules, Build status config

Git clone depth - optional

1 ▾

Git submodules - optional

Use Git submodules

Build Status - optional

Report build statuses to source provider when your builds start and finish

Status context - optional

Configurable context with build status with support for environment variables.

- I entered the GitHub repository URL (github.com in Bing).
- This connects the CodeBuild project to the source code for the FastFood application.
- The repository field confirms the link to the correct GitHub project.

Operating system  
Ubuntu

Runtime(s)  
Standard

Image  
aws/codebuild/standard:7.0

Image version  
Always use the latest image for this runtime version

Use GPU-enhanced compute

Service role

New service role  
Create a service role in your account

Existing service role  
Choose an existing service role from your account

Role ARN  
arn:aws:iam::586549103866:role/CodeBuild-Service-Role

Allow AWS CodeBuild to modify this service role so it can be used with this build project.

① Select Ubuntu  
② Select Standard  
③ Select this  
④ Select this  
⑤ Choose this  
⑥ It will automatically give your created role

- I selected Ubuntu as the operating system for the build environment.
- I chose Standard as the runtime, which supports a wide range of build tools and languages.
- For the image, I selected , which includes pre-installed runtimes like Python, Node.js, and Java.
- I set the image version to “Always use the latest image for this runtime version” to ensure the build uses updated tools.
- I did not enable GPU-enhanced compute since it's not needed for this application.
- For the service role, I selected “Existing service role” and chose from the account.
- I allowed CodeBuild to modify the service role so it can be used with this build project.
- This completes the environment setup for the build process.

## ▼ Buildspec

### Build specifications

#### Insert build commands

Store build commands as build project configuration

#### Use a buildspec file

Store build commands in a YAML-formatted buildspec file

### 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).

buildspec.yml



- I selected “Use a buildspec file” to define build commands in YAML format.
- This allows the build instructions to be stored inside the source code repository.
- I entered the filename which is the default name CodeBuild looks for in the root directory.

Developer Tools > CodeBuild > Build projects > Create build project

Amazon S3

You might choose no artifacts if you are running tests or pushing a Docker image to Amazon ECR.

Bucket name

fastfood-cicd-artifacts-vighnesh1

Name

The name of the folder or compressed file in the bucket that will contain your output artifacts. Use Artifacts packaging under Additional configuration to choose whether to use a folder or compressed file. If the name is not provided, defaults to project name.

Enable semantic versioning  
Use the artifact name specified in the buildspec file

Path - optional

The path to the build output ZIP file or folder.

Example: MyPath/MyArtifact.zip.

Namespace type - optional

None

Choose Build ID to insert the build ID into the path to the build output ZIP file or folder, e.g. MyPath/MyBuildID/MyArtifact.zip. Otherwise, choose None.

Artifacts packaging

None  
The artifact files will be uploaded to the bucket.

Zip  
AWS CodeBuild will upload artifacts into a compressed file that is put into the specified bucket.

① Select Amazon S3  
② Select your bucket  
③ Select the zip

This screenshot shows the 'Create build project' wizard in the AWS CodeBuild console. The 'Artifacts packaging' section is highlighted with handwritten annotations. Step 1, 'Select Amazon S3', points to the storage dropdown. Step 2, 'Select your bucket', points to the bucket input field. Step 3, 'Select the zip', points to the 'Zip' radio button under 'Artifacts packaging'.

- I selected “Use a buildspec file” to define build commands in YAML format.
- This allows CodeBuild to read instructions directly from the source repository.
- I entered the file name as which is the default name CodeBuild looks for in the root directory.
- This file will contain phases like install, build, and post\_build, along with artifact definitions.
- Using a buildspec file keeps the build logic version-controlled and easy to update.

**▼ Logs**

CloudWatch

 **CloudWatch logs - optional**

Checking this option will upload build output logs to CloudWatch.

**Group name - optional**

aws/codebuild/FastFood-Build

The group name of the logs in CloudWatch Logs. The log group name will be /aws/codebuild/&lt;project-name&gt; by default.

**Stream name prefix - optional**

The prefix of the stream name of the CloudWatch Logs.

S3

 **S3 logs - optional**

Checking this option will upload build output logs to S3.

[Cancel](#)[Create build project](#)*① Click here*

- I selected Amazon S3 as the destination for storing build artifacts.
- This allows the output of the build process to be saved in a centralized, versioned location.
- I chose the bucket named which was created earlier for this purpose.
- For packaging, I selected the “Zip” option so that all build output files are compressed into a single archive.
- This makes it easier to transfer and deploy the artifacts using CodeDeploy.

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, and Capacity Manager. Below these are Images, AMIs, AMI Catalog, and Elastic Block Store sections. The main area shows a table with one instance: FastFood-CICD... (Instance ID: i-0930920d7c6557ceb, State: Running, Type: t3.micro). A tooltip from the Actions dropdown is open, listing options: Change security groups, Get Windows password, and Modify IAM role. Handwritten annotations include: '① Go to instances' pointing to the Instances link in the sidebar; '② check this box' pointing to the checkbox in the table header; '③ Select actions' pointing to the Actions dropdown; '④ select security' pointing to the Security option in the Actions dropdown; and '⑤ Modify IAM role' pointing to the Modify IAM role option in the tooltip.

- I opened the EC2 console and navigated to the Instances section.
- From the Actions dropdown, I hovered over Security.
- I chose Modify IAM role to attach the correct permissions for CodeDeploy and CodeBuild access.

## Modify IAM role Info

Attach an IAM role to your instance.

### Instance ID

i-0930920d7c6557ceb (FastFood-CICD-Server)

### IAM role

Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

EC2-CodeDeploy-Role



Create new IAM role

Cancel

Update IAM role

① Select this role

② Click on Update IAM role

- I opened the Modify IAM role section for the EC2 instance named FastFood-CICD-Server.
- I selected the IAM role named EC2-CodeDeploy-Role from the dropdown.
- This role includes permissions for CodeDeploy to interact with the EC2 instance during deployments.
- I clicked on Update IAM role to attach it to the instance.
- This step ensures the EC2 instance can receive and execute deployment instructions securely.

The screenshot shows the AWS Console Home page. A handwritten note at the top right says "① click codeDeploy". A hand-drawn arrow points from this note to the "CodeDeploy" service icon in the "Recently visited" sidebar. The "CodeDeploy" service card is highlighted with a yellow box. The main content area shows the "Applications" section with a message: "No applications. Get started by creating an application." A "Create application" button is visible. Other sections like "Welcome to AWS", "AWS Health", and "Cost and usage" are also present.

Console Home [Info](#)

① click codeDeploy

Reset to default layout + Add widgets

Recently visited [Info](#)

- EC2
- IAM
- S3
- CodePipeline
- CodeBuild
- VPC
- Support
- Billing and Cost Management

CodeDeploy

Service Quotas

Simple Notification Service

CloudWatch

Applications (0) [Info](#)

Region: US East (N. Virginia)

Select Region: us-east-1 (Current Region) [▼](#)

Find applications

Name | Description | Region | Originati. [▼](#) [▲](#)

No applications  
Get started by creating an application.  
[Create application](#)

View all services

Go to myApplications

Welcome to AWS

Getting started with

AWS Health [Info](#)

Open issues

Cost and usage [Info](#)

Current month Cost (\$)

© 2026, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

- I opened the AWS Console and navigated to the CodeDeploy service.
- In the Applications section, I clicked on “Create application” to begin the deployment setup.
- This application will manage deployments for the FastFood project.

The screenshot shows the AWS CodeDeploy Applications page. On the left, a sidebar menu lists various services under 'Developer Tools' and 'CodeDeploy'. The 'Applications' option is highlighted with a blue arrow and the handwritten note '(1) click on applications'. In the main content area, the 'Applications' section is displayed with columns for 'Application name', 'Compute platform', and 'Created'. A search bar is at the top, followed by a message 'No results' and 'There are no results to display.' At the top right, there are buttons for 'Notify', 'View details', 'Deploy application', and a prominent orange 'Create application' button. A large handwritten note '(2) click on Create application' with an arrow points to the orange button.

- I clicked on “Create application” to start a new deployment configuration.
- This application will manage how build artifacts are delivered to the EC2 instance.
- Creating the application is the first step before defining deployment groups and specifying deployment behavior.
- The application name will match the project it supports, keeping the CI/CD structure organized.



## Create application

Application configuration

Application name  
Enter an application name  
 100 character limit

Compute platform  
Choose a compute platform  
 ▾

Tags

Cancel  2

① Give the Application name  
② Select EC2/On-premises  
③ Click here

- I entered the application name which matches the project name for clarity.
- I selected the compute platform since the deployment target is an EC2 instance.
- This platform supports deployments to virtual machines managed by AWS.
- I left the tags section empty for now, as tagging is optional.
- Then I clicked “Create application” to finalize the setup.
- This completes the application creation step in CodeDeploy.

The screenshot shows the AWS CodeDeploy console. On the left, a sidebar menu includes options like Source, Artifacts, Build, Deploy (selected), Pipeline, and Settings. Under Deploy, 'Application' is expanded, showing Settings, Deployment configurations, and On-premises instances. Below these are Pipeline and Settings. At the bottom of the sidebar are links for Go to resource and Feedback.

The main content area has a green header bar with the message: "Application created. In order to create a new deployment, you must first create a deployment group." Below this is a form with fields for Name (FastFood-App) and Compute platform (EC2/On-premises). The tabs at the top of the main content area are Deployments, Deployment groups (selected), and Revisions. The Deployment groups section contains a search bar, buttons for View details, Edit, and Create deployment group (which is highlighted with a yellow box and a handwritten note "Click here"), and a pagination control showing page 1 of 1. Below this is a table with columns for Name, Status, Last attempted deploy..., Last successful deploy..., and Trigger count. The table currently displays "No deployment groups". A message below the table states: "Before you can deploy your application using CodeDeploy, you must create a deployment group." A "Create deployment group" button is located at the bottom of this message.

- I navigated to the Deployment groups tab, which showed no existing groups.
- I clicked on “Create deployment group” to begin configuring how deployments will be delivered to the EC2 instance.
- This step is required before any application revision can be deployed.
- The deployment group will define the target instance, deployment settings, and service role permissions.

Enter a deployment group name  
FastFood-Deployment-Group  
100 character limit

① Drive the name

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::586549103866:role/CodeDeploy-Service-Role

② select the service role

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.

③ select In-place

- I entered the deployment group name as FastFood-Deployment-Group, which clearly identifies the group linked to the FastFood-App.
- I selected the service role CodeDeploy-Service-Role, which includes permissions for CodeDeploy to manage deployments on EC2 instances.
- For deployment type, I chose In-place, which updates the existing EC2 instance with the latest application revision.

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  Remove tag

Add tag [+ Add tag group](#)

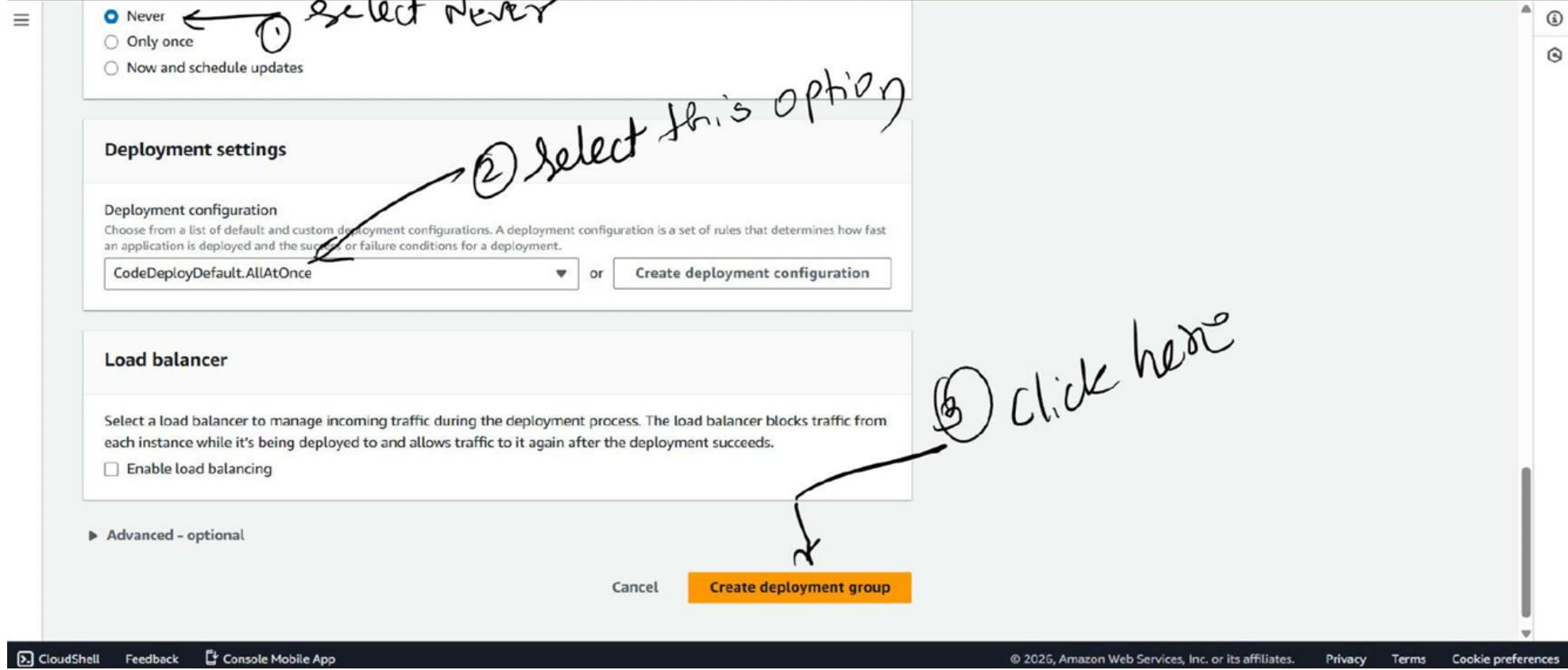
On-premises instances

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

① Check this option  
② Select Name  
③ Select this  
④ This sentence must appear

The screenshot shows the AWS CodeDeploy target selection interface. It highlights the 'Amazon EC2 instances' checkbox with a circled '1'. Handwritten text 'Check this option' points to it. Below, a 'Tag group 1' section is shown with a 'Name' key and a 'FastFood-CICD-Server' value, with a circled '2' next to 'Select Name'. To the right, handwritten text 'Select this' points to the 'Multiple tag groups' explanatory text. At the bottom, handwritten text 'This sentence must appear' points to the 'Matching instances' section, which displays '1 unique matched instance' with a circled '4'.

- I selected “Amazon EC2 instances” as the deployment target type.
- This allows CodeDeploy to deliver application revisions directly to the EC2 instance.
- I added a tag group with the key set to Name and the value set to FastFoodp-CICD-Server .
- This tag matches the EC2 instance created earlier for hosting the application.
- The interface confirmed “1 unique matched instance,” which verifies that the tag correctly identifies the target.



- I selected “Never” for scheduling updates, meaning deployments will only occur when manually triggered.
- For deployment configuration, I chose `CodeDeployDefaultAllAtOnce`, which updates all instances in the group simultaneously.
- This method is faster and suitable for single-instance deployments like FastFood-CI-CD-Server.
- I did not enable load balancing since the deployment targets only one EC2 instance and does not require traffic routing.
- Finally, I clicked “Create deployment group” to complete the setup.

The screenshot shows the AWS Console Home page. On the left, there's a sidebar with a hand-drawn annotation pointing to the 'CodePipeline' item under 'Recently visited'. The main area has four main sections: 'Recently visited', 'Applications', 'Welcome to AWS', and 'AWS Health'. The 'Applications' section is currently active, showing a table with no applications and a 'Create application' button. The 'Welcome to AWS' section includes links for 'Getting started with' and 'CloudShell'. The 'AWS Health' section shows 'Open issues'. At the bottom, there are links for 'Feedback', 'Console Mobile App', and 'Cookie preferences'.

Recently visited

- CodeDeploy
- EC2
- IAM
- S3
- CodePipeline
- CodeBuild
- VPC
- Support

Billing and Cost Management

Service Quotas

Simple Notification Service

CloudWatch

Applications (0)

Region: US East (N. Virginia)

Select Region: us-east-1 (Current Region)

Create application

No applications  
Get started by creating an application.

Welcome to AWS

Getting started with

AWS Health

Cost and usage

Current month

Cost (\$)

CloudShell Feedback Console Mobile App © 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- Launching CodePipeline allows me to define the stages of the pipeline, including source retrieval, build execution, and deployment delivery.
- This step initiates the creation of a continuous integration and deployment process for the FastFood project.

Pipelines Info

No results  
There are no results to display.

Click on Create pipeline

Create pipeline

Name	Latest execution status	Latest source revisions	Latest execution started	Most recent executions
------	-------------------------	-------------------------	--------------------------	------------------------

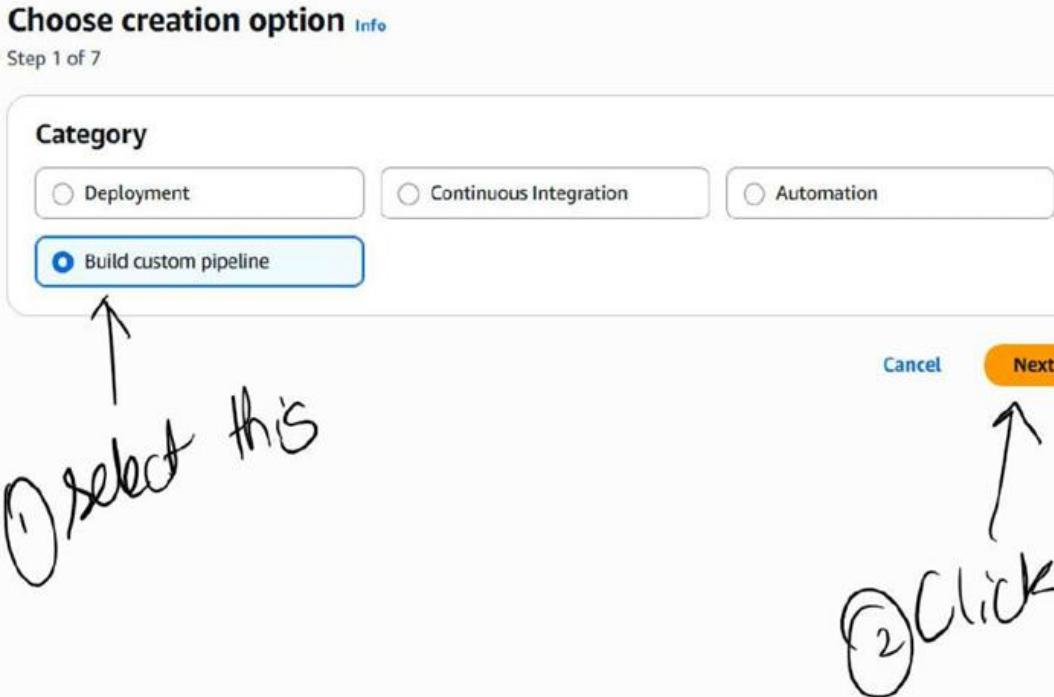
View history Release change Delete pipeline

CloudShell Feedback Console Mobile App

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- I opened the AWS Console and navigated to the CodePipeline service.
- In the Pipelines tab, I clicked on Create pipeline to begin setting up the CI/CD workflow.
- This pipeline will automate the process by connecting CodeCommit, CodeBuild, and CodeDeploy.

- Step 1  
**Choose creation option**
- Step 2  
Choose pipeline settings
- Step 3  
Add source stage
- Step 4 - optional  
Add build stage
- Step 5 - optional  
Add test stage
- Step 6  
Add deploy stage
- Step 7  
Review



- I entered the deployment group name as FastFood-Deployment-Group.
- I selected the service role CodeDeploy-Service-Role, which grants AWS CodeDeploy access to the EC2 instance.
- For deployment type, I chose In-place, which updates the existing EC2 instance with the latest application revision.
- This method briefly takes the instance offline during deployment but avoids provisioning new infrastructure.

- Step 2  
Choose pipeline settings
- Step 3  
Add source stage
- Step 4 - optional  
Add build stage
- Step 5 - optional  
Add test stage
- Step 6  
Add deploy stage
- Step 7  
Review

**Pipeline settings**

**Pipeline name**  
Enter the pipeline name. You cannot edit the pipeline name after it is created.  
 → ① Give the pipeline name

No more than 100 characters

**Execution mode** Info  
Choose the execution mode for your pipeline. This determines how the pipeline runs.  
 Superseded  
 Queued → ② Select Queued  
 Parallel

**Service role**  
 New service role → ③ Select 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

**▼ Advanced settings**

- I entered the pipeline name as FastFood-Pipeline.
- For execution mode, I selected Queued, which allows multiple executions to run in sequence.
- I chose New service role so that AWS CodePipeline can automatically create a role with the required permissions.
- The role name generated was AWSCodePipelineServiceRole-us-east-1-FastFood-Pipeline.

The screenshot shows the 'Create new pipeline' wizard in the AWS CodePipeline console. The current step is 'Set artifact store and encryption key'. The 'Artifact store' section has 'Default location' selected, which creates a default S3 bucket in the account. The 'Encryption key' section also has 'Default AWS Managed Key' selected, using the built-in customer master key for the pipeline. Below these sections is a 'Variables' section where users can add variables at the pipeline level. A handwritten note with a circled '1' and the text 'click on next' points to the 'Next' button at the bottom right of the page.

Artifact store

Default location  
Create a default S3 bucket in your account.

Custom location  
Choose an existing S3 location from your account in the same region and account as your pipeline

Encryption key

Default AWS Managed Key  
Use the AWS managed customer master key for CodePipeline in your account to encrypt the data in the artifact store.

Customer Managed Key  
To encrypt the data in the artifact store under an AWS KMS customer managed key, specify the key ID, key ARN, or alias ARN.

**Variables**

You can add variables at the pipeline level. You can choose to assign the value when you start the pipeline. [Learn more](#)

No variables defined at the pipeline level in this pipeline.

Add variable

You can add up to 50 variables.

Cancel Previous Next

① Click on next

- I selected the default location for the artifact store, allowing AWS to create a default S3 bucket for storing pipeline artifacts.
- For encryption, I chose the default AWS Managed Key, which uses the built-in customer master key for CodePipeline.

- Step 1  
Choose creation option
- Step 2  
Choose pipeline settings
- Step 3  
**Add source stage**
- Step 4 - optional  
Add build stage
- Step 5 - optional  
Add test stage
- Step 6  
Add deploy stage
- Step 7  
Review

### Add source stage Info

Step 3 of 7

**Source**

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

GitHub (via GitHub App) arn:aws:codeconnections:us-east-1:123456789012:connection/12345678901234567890123456789012 or [Connect to GitHub](#)

**Connection**  
Choose an existing connection that you have already configured, or create a new one and then return to this task.

or [Connect to GitHub](#)

**Repository name**  
Choose a repository in your GitHub account.

X

You can type or paste the group path to any project that the provided credentials can access. Use the format: 'group/subgroup/project'.

**Default branch**  
Default branch will be used only when pipeline execution starts from a different source or manually started.

X

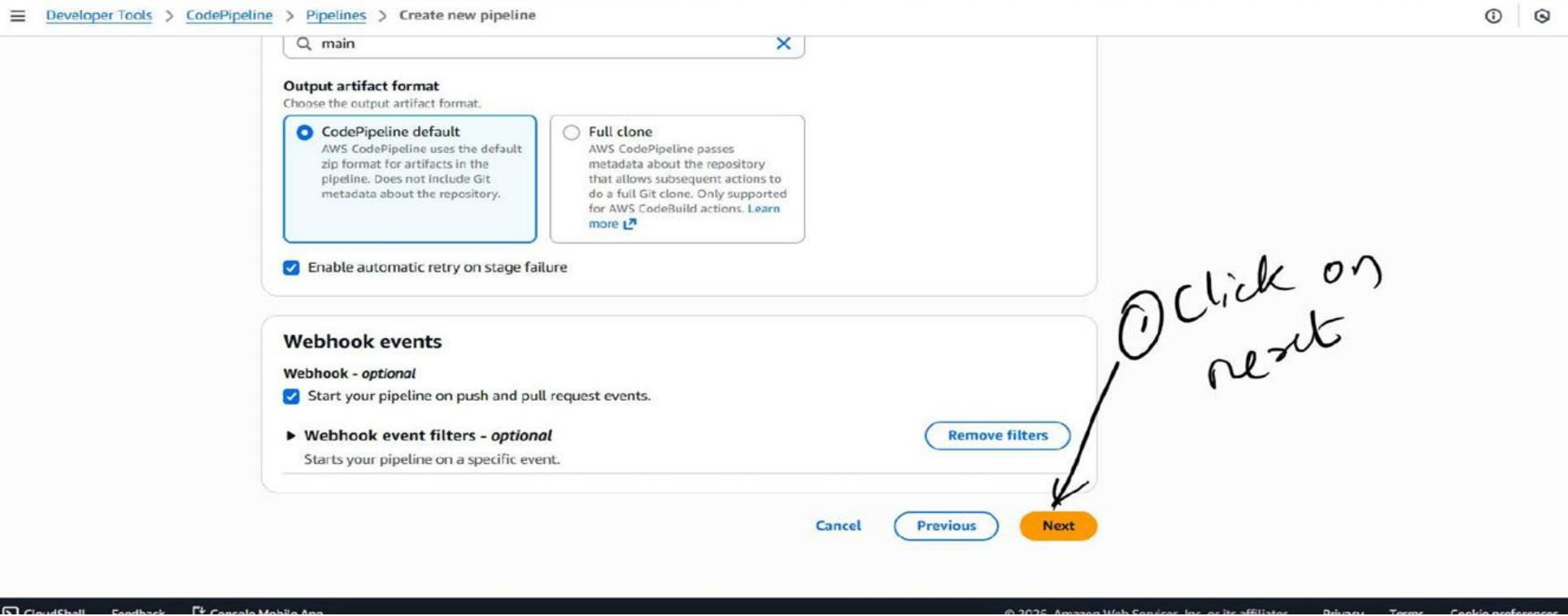
Choose the output artifact format:

**CodePipeline default**  
AWS CodePipeline uses the default

**Full clone**  
AWS CodePipeline passes

*① select GitHub*  
*② choose the same connection one used for Code build*  
*③ choose the repo*  
*④ select main and scroll down?*

- I selected GitHub (via GitHub App) as the source provider.
- I chose the same connection used earlier for CodeBuild to maintain consistency.
- For repository name, I selected vighnesh32/fastfood-cicd-project.
- I set the default branch to main, which will be used for triggering pipeline executions.



- I selected CodePipeline default as the output artifact format, which stores artifacts in zip format without Git metadata.
- I enabled automatic retry on stage failure to improve reliability during pipeline execution.
- I checked the webhook option to allow the pipeline to start automatically on push and pull request events from GitHub.

- Step 1 Choose creation option
- Step 2 Choose pipeline settings
- Step 3 Add source stage
- Step 4 - optional Add build stage
- Step 5 - optional Add test stage
- Step 6 Add deploy stage
- Step 7 Review

### Add build stage - optional Info

Step 4 of 7

**Build - optional**

**Build provider**

Choose the tool you want to use to run build commands and specify artifacts for your build action.

Commands  Other build providers

AWS CodeBuild

**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.

FastFood-Build x or [Create project](#)

Define buildspec override - optional

Buildspec file or definition that overrides the latest one defined in the build project, for this build only.

**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**

1 select this option  
2 Select AWS CodeBuild  
3 select existing project

- I selected Other build providers and chose AWS CodeBuild as the build provider.
- For the project name, I selected FastFood-Build, which was already created in the CodeBuild console.
- I did not define a buildspec override or add environment variables at this stage.

Developer Tools > CodePipeline > Pipelines > Create new pipeline

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  
Triggers a single build.

Batch build  
Triggers multiple builds as a single execution.

Region

United States (N. Virginia)

Input artifacts

Choose an input artifact for this action. [Learn more](#)

SourceArtifact X  
Defined by: Source

Enable automatic retry on stage failure

Cancel Previous Skip build stage Next

① select single build →

② choose the region

③ click on Next

CloudShell Feedback Console Mobile App

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- I selected Single build as the build type, which triggers one build per pipeline execution.
- For region, I chose United States (N. Virginia) to match the location of the EC2 instance and other resources.

- Step 1  
[Choose creation option](#)
- Step 2  
[Choose pipeline settings](#)
- Step 3  
[Add source stage](#)
- Step 4 - optional  
[Add build stage](#)
- Step 5 - optional  
[Add test stage](#)
- Step 6  
[Add deploy stage](#)
- Step 7  
[Review](#)

## Add test stage - optional info

Step 5 of 7

### Test - optional

#### Test provider

Choose how you want to test your application or content. Choose the provider, and then provide the configuration details for that provider.



Enable automatic retry on stage failure

[Cancel](#)[Previous](#)[Skip test stage](#)[Next](#)

① Click on next

- I enabled automatic retry on stage failure to ensure resilience in case of transient issues.
- Then I clicked Next to proceed to the deploy stage configuration.

Screenshot of the AWS CodePipeline 'Create new pipeline' step 7: Add deploy stage.

The page shows the configuration for a deployment stage:

- Deploy provider:** AWS CodeDeploy
- Region:** United States (N. Virginia)
- Input artifacts:** BuildArtifact (Defined by: Build)
- Application name:** FastFood-App
- Deployment group:** FastFood-Deployment-Group

Handwritten notes and numbered steps:

- ① select this option
- ② choose your region
- ③ It will come automatically
- ④ put the application name
- ⑤ Select the deployment group

At the bottom, there are checkboxes for "Configure automatic rollback on stage failure" (checked) and "Enable automatic retry on stage failure".

- I selected AWS CodeDeploy as the deploy provider.
- For region, I chose United States (N. Virginia) to match the location of the application and EC2 instance.
- The input artifact BuildArtifact was automatically selected from the build stage.
- I chose the application name FastFood-App, which was created earlier in the CodeDeploy console.
- For deployment group, I selected FastFood-Deployment-Group to target the correct EC2 instance.

Enable automatic retry on stage failure  
Enabled

### Step 6: Add deploy stage

#### Deploy action provider

Deploy action provider  
AWS CodeDeploy

ApplicationName  
FastFood-App

DeploymentGroupName  
FastFood-Deployment-Group

Configure automatic rollback on stage failure  
Enabled

Enable automatic retry on stage failure  
Disabled

Once and  
click on  
Create pipeline



review



Cancel

Previous

Create pipeline

- I reviewed the deploy stage configuration with AWS CodeDeploy as the deploy action provider.

## FastFood-Pipeline

Pipeline

Executions

Triggers

Settings

Tags

Stage

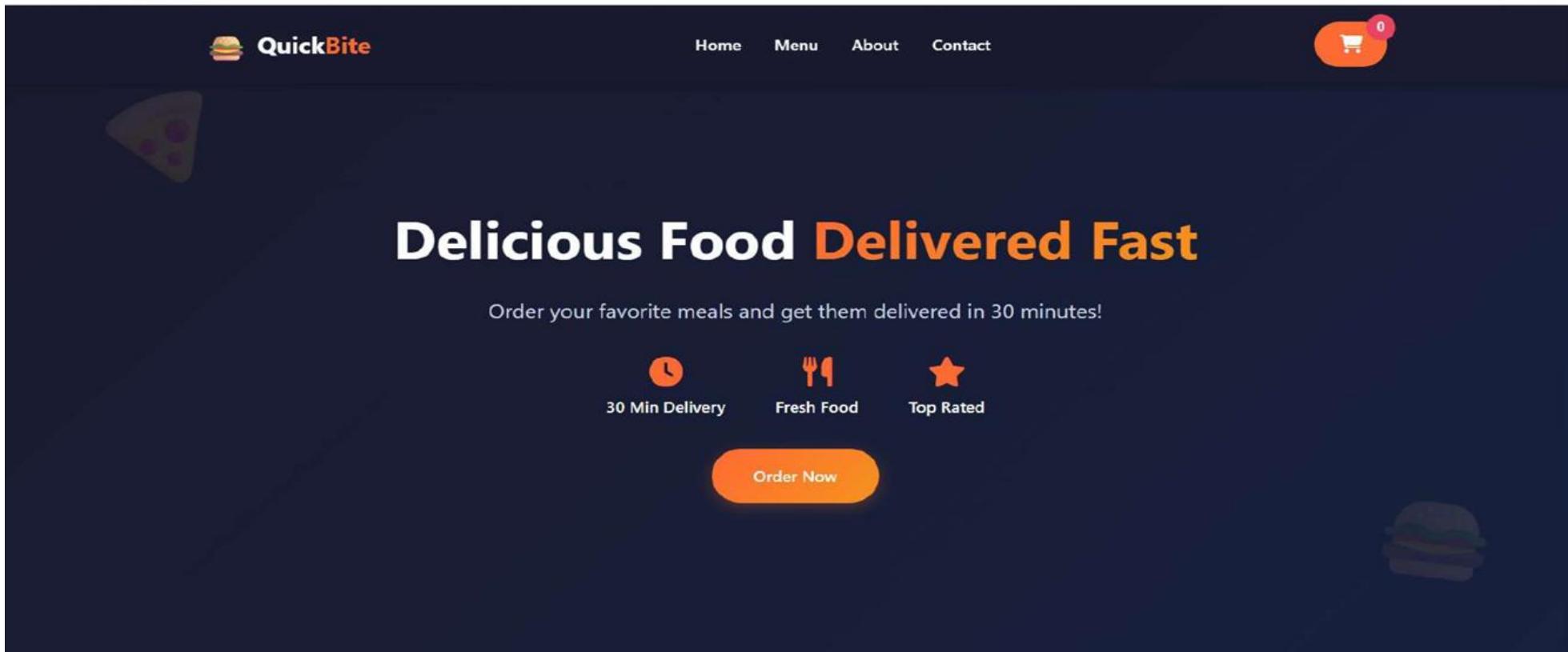
The CI/CD pipeline created successfully



Horizontal Vertical



- The Build stage compiled the application using AWS CodeBuild.
- The Deploy stage delivered the application to the EC2 instance using AWS CodeDeploy.
- All actions succeeded, confirming that the CI/CD pipeline is functioning correctly.



- After successful deployment, I verified the frontend by accessing the QuickBite web interface.

A screenshot of a web browser window. The address bar shows the URL "44.199.191.251:3000/health". The page content is a JSON object with the following structure:

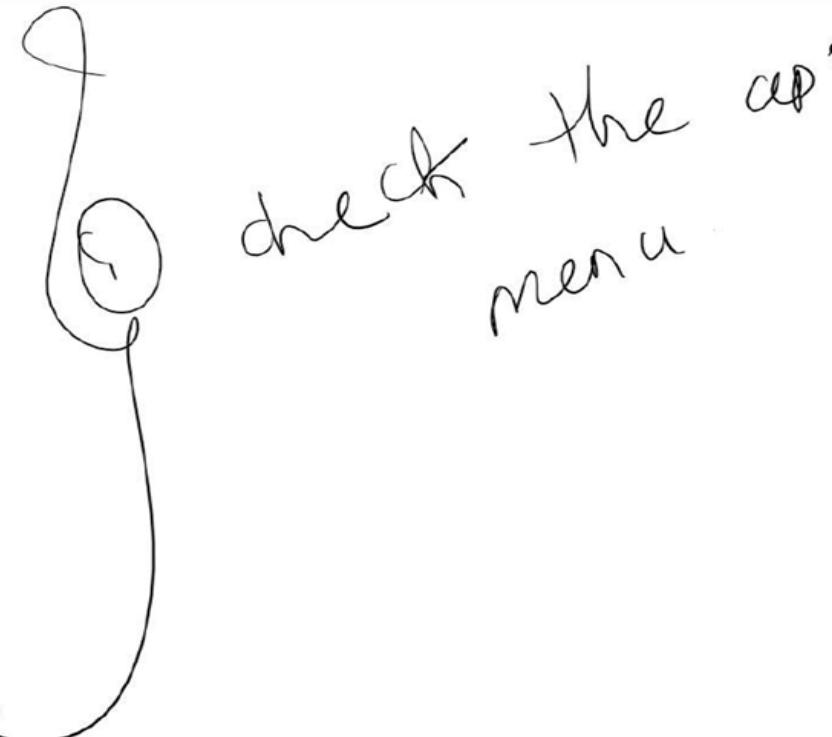
```
{  
  "status": "healthy",  
  "timestamp": "2026-01-02T16:01:27.715Z",  
  "version": "1.0",  
  "menuItems": 16,  
  "message": "Fast Food App is running smoothly!"  
}
```

Handwritten text "go check the health" is written across the top right of the JSON output.

- I verified the backend by accessing the health check endpoint
- The JSON response confirmed the application status as healthy with the message “Fast Food App is running smoothly!”

netty-print

```
{  
    "id": 1,  
    "name": "Classic Burger",  
    "category": "burgers",  
    "price": 299,  
    "description": "Juicy aloo patty with lettuce, tomato, and special sauce",  
    "image": "🍔"  
},  
{  
    "id": 2,  
    "name": "Cheese Burger",  
    "category": "burgers",  
    "price": 349,  
    "description": "Double cheese with premium aloo patty",  
    "image": "🧀"  
},  
{  
    "id": 3,  
    "name": "Chicken Burger",  
    "category": "burgers",  
    "price": 279,  
    "description": "Crispy chicken fillet with mayo and lettuce",  
    "image": "🐔"  
},  
{  
    "id": 4,  
    "name": "Veggie Burger",  
    "category": "burgers",  
    "price": 249,  
    "description": "Plant-based patty with fresh vegetables",  
    "image": "🥕"  
},  
{  
    "id": 5,  
    "name": "Margherita Pizza",  
    "category": "pizza",  
    "price": 399,  
    "description": "Classic cheese pizza with fresh basil",  
    "image": "🍕"  
},  
{  
    "id": 6,
```



- I verified the API endpoint at (44.199.191.251 in Bing) to confirm menu data delivery.
- The response returned a valid JSON array containing multiple menu items with fields like id, name, category, price, description, and emoji.
- Sample items included Classic Burger, Cheese Burger, Chicken Burger, Veggie Burger, and Margherita Pizza.
- Each item was correctly categorized and priced, confirming that the backend is serving structured data as expected.

