# AWS MERN Stack Deployment Guide

## Step 1: Launch an EC2 Instance

1. Log in to AWS Management Console.
2. Navigate to EC2 Dashboard → Click Launch Instance.
3. Choose Ubuntu 22.04 LTS (recommended) as your OS.
4. Select t2.micro (free-tier eligible) or a higher instance if needed.
5. Click Next: Configure Instance Details → Keep default settings.
6. Click Next: Add Storage → Increase root volume (at least 20GB recommended).
7. Click Next: Add Tags → Add a tag (optional, e.g., Name = MERN-Server).
8. Click Next: Configure Security Group:
   * Add rules:
     + SSH (22) → Source: My IP
     + HTTP (80) → Source: Anywhere (0.0.0.0/0)
     + HTTPS (443) → Source: Anywhere (0.0.0.0/0)
     + Custom TCP (3000, 5000, etc.) → Source: Anywhere (0.0.0.0/0)
9. Click Review and Launch.
10. Create and download a new key pair (.pem file) → Launch Instance.

## Step 2: Connect to EC2 via PuTTY

1. Convert .pem to .ppk for PuTTY:
   * Open PuTTYgen → Load .pem file → Save Private Key (.ppk).
2. Open PuTTY and connect:
   * Host Name: ubuntu@your-ec2-public-ip (for Ubuntu instances)
   * Port: 22
   * Auth: Load the .ppk file under "SSH → Auth → Credentials"
   * Click Open.

**Common Errors:**

* Connection timeout: Ensure SSH (22) is allowed in Security Group.
* Server refused key: Use correct .ppk file.
* Permission denied: Change key permissions (chmod 400 key.pem in Linux/Mac).

## Step 3: Update & Install Required Packages

Run the following in PuTTY:

sudo apt update && sudo apt upgrade -y

### Install Node.js and npm

# Remove existing Node.js and npm (if any)

sudo apt remove --purge nodejs npm -y

sudo apt autoremove -y

# Install Node.js from NodeSource

curl -fsSL https://deb.nodesource.com/setup\_18.x | sudo -E bash -

sudo apt install -y nodejs

# Verify installation

node -v

npm -v

### Install MongoDB Community Edition

1. Import the public key:

sudo apt-get install gnupg curl -y

curl -fsSL https://www.mongodb.org/static/pgp/server-8.0.asc | \

sudo gpg -o /usr/share/keyrings/mongodb-server-8.0.gpg \

--dearmor

1. Create the list file (for Ubuntu 22.04 Jammy):

echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-8.0.gpg ] https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/8.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-8.0.list

1. Reload the package database:

sudo apt-get update

1. Install MongoDB Community Server:

sudo apt-get install -y mongodb-org

1. Start MongoDB and enable on boot:

sudo systemctl start mongod

sudo systemctl enable mongod

sudo systemctl status mongod

**Common Errors:**

* MongoDB failed to start: Run sudo journalctl -xe for logs.
* Node.js not found: Check which node or re-install Node.js.

## Step 4: Transfer MERN Project Using FileZilla

1. Open FileZilla.
2. Go to Edit → Settings → SFTP → Add key file (.pem or .ppk).
3. In FileZilla Site Manager:
   * Host: your-ec2-public-ip
   * Protocol: SFTP
   * User: ubuntu (for Ubuntu EC2 instances)
   * Password: Leave blank
   * Logon Type: Key File
   * Key File: Select .ppk
   * Click Connect.
4. Upload your MERN project to /home/ubuntu/project.

**Common Errors:**

* Permission denied: Use sudo chmod -R 755 /home/ubuntu/project
* Cannot connect: Ensure EC2 Public IP and SFTP rule (22) are correct.

## Step 5: Install Dependencies & Start Backend

1. Navigate to your project:

cd /home/ubuntu/project/backend

1. Install dependencies:

npm install

1. Start backend using PM2 (production process manager):

# Install PM2

npm install -g pm2

# Start the server with PM2

pm2 start server.js

# Save the PM2 process list

pm2 save

# Configure PM2 to start on system boot

pm2 startup

**Common Errors:**

* Module not found: Run npm install
* Port already in use: Use lsof -i :5000 to check and kill -9 PID to free the port

## Step 6: Deploy Frontend

1. Navigate to frontend folder:

cd /home/ubuntu/project/frontend

1. Install dependencies:

npm install

1. Build project:

npm run build

1. Install & configure NGINX:

sudo apt install nginx -y

sudo nano /etc/nginx/sites-available/default

Replace the default configuration with:

server {

listen 80;

server\_name your-ec2-public-ip;

location / {

root /home/ubuntu/project/frontend/build;

index index.html;

try\_files $uri /index.html;

}

location /api/ {

proxy\_pass http://localhost:5000/;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

}

}

Save and restart NGINX:

sudo systemctl restart nginx

sudo systemctl status nginx

## Step 7: Enable HTTPS (SSL)

1. Install Certbot:

sudo apt install certbot python3-certbot-nginx -y

1. Obtain and configure SSL certificate:

sudo certbot --nginx -d your-domain.com

1. Test auto-renewal:

sudo certbot renew --dry-run

## Alternative: Deploying a Next.js Application

### For Next.js Projects

1. Navigate to your project:

cd /home/ubuntu/project/nextjs-app

1. Install dependencies:

npm install

1. Build the Next.js application:

npm run build

1. Start Next.js in production mode with PM2:

pm2 start npm --name "nextjs" -- start

pm2 save

1. Configure NGINX for Next.js:

sudo nano /etc/nginx/sites-available/default

Replace with:

server {

listen 80;

server\_name your-ec2-public-ip;

location / {

proxy\_pass http://localhost:3000;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

}

}

### For Vite-based Development Servers

If you need to expose your development server on port 5173:

1. Open port 5173 in AWS Security Group:
   * Go to AWS EC2 Dashboard
   * Select your EC2 instance
   * Click on the Security Groups
   * Under the Inbound rules tab, click Edit inbound rules
   * Add Rule: Custom TCP (5173), Source: 0.0.0.0/0
   * Click Save rules
2. Run Vite with public access:

npm run dev -- --host 0.0.0.0

1. Access the app at: http://your-ec2-public-ip:5173/

**Note:** For production, build the application and serve it through NGINX instead of exposing the development server.

## Common Debugging Issues

* NGINX Configuration Error: Run sudo nginx -t to check config
* MongoDB connection failed: Ensure MongoDB is running (sudo systemctl status mongod) and update connection strings in your application
* Frontend not loading: Check that the build folder exists and NGINX has proper permissions
* Security issues: Ensure firewall allows necessary ports (sudo ufw status)
* Application crashes: Check PM2 logs with pm2 logs