Software Quality Engineering Project Documentation

Comprehensive Quality Engineering for Nextcloud Server

Project Overview

Application Selected: Nextcloud Server (Open-source file synchronization and sharing platform)
Repository: https://github.com/nextcloud/server
CI/CD Tool: Jenkins
Deployment Platform: AWS CodeDeploy + EC2
Testing Frameworks: PHPUnit (Backend), Cypress (Frontend)

---

1. Source Stage: Code Repository & Pipeline Triggering

1.1 Repository Setup

Forking and Cloning:

```
# Fork nextcloud/server repository on GitHub
# Clone the forked repository locally
git clone git@github.com:<username>/nextcloud-server.git
cd nextcloud-server

# Create development branch
git switch -c develop
git push -u origin develop
```

Branch Protection Configuration:

- Protected branches: `main` and `develop`
- Required: Pull request reviews before merging
- Required: Status checks must pass before merging
- Linear history enforcement enabled

1.2 Jenkins Installation and Configuration

Jenkins Setup on Windows:

```
# Install Jenkins on Windows
# Access Jenkins at http://localhost:8080
# Complete initial setup wizard with admin password
```

Required Jenkins Plugins:

- Git Plugin

- GitHub Branch Source Plugin
- Pipeline Plugin
- Pipeline: Multibranch Plugin
- Credentials Binding Plugin

Plugin Installation Steps:

1. Navigate to: Manage Jenkins → Manage Plugins → Available
2. Search and install required plugins
3. Restart Jenkins after installation

1.3 GitHub Credentials Configuration

Creating GitHub Personal Access Token (PAT):

1. GitHub Settings → Developer settings → Personal access tokens → Generate token
2. Required scopes: `repo`, `admin:repo_hook`
3. Save token securely

Adding Credentials to Jenkins:

1. Jenkins → Credentials → System → Global credentials → Add Credentials
2. Kind: Username with password
3. Username: GitHub username
4. Password: Generated PAT
5. ID: `github-credentials`

1.4 Multibranch Pipeline Setup

Creating Jenkins Job:

1. Jenkins Dashboard → New Item → Enter name: `nextcloud-server`
2. Select: Multibranch Pipeline → OK
3. Configuration:
   - Branch Sources → Add source → GitHub
   - Repository URL: `https://github.com/<username>/nextcloud-server.git`
   - Credentials: Select `github-credentials`
   - Behaviors:
     - Discover branches
     - Discover pull requests from origin
     - Filter by name (regex): `^(main|develop|feature/.*)$`

Shallow Clone Configuration:

- Add behavior: Additional clone options
- Enable shallow clone with depth: 1
- Timeout: 60 minutes

- This addresses large repository size issues

1.5 Repository Structure

```
nextcloud-server/
│
├── appspec.yml                    # AWS CodeDeploy specification
├── Jenkinsfile                    # Pipeline configuration
├── scripts/
│   ├── install_dependencies.sh    # Dependency installation script
│   ├── start_server.sh           # Server startup script
│   └── stop_server.sh            # Server shutdown script
│
├── frontend_tests/               # Cypress UI tests
│   ├── cypress.config.js
│   └── tests/
│
├── backend_tests/                # PHPUnit backend tests
│   ├── phpunit.xml
│   └── tests/
│
├── src/                          # Application source code
├── composer.json                 # PHP dependencies
├── package.json                  # Node.js dependencies
└── README.md
```

1.6 Git Configuration for Large Repositories

To handle the large Nextcloud repository size, the following Git configurations were applied:

```
git config --global http.postBuffer 524288000
git config --global http.maxRequestBuffer 1000m
git config --global core.compression 0
git config --global http.sslBackend schannel
git config --global http.lowSpeedLimit 0
git config --global http.lowSpeedTime 999999
```

---

2. Build Stage: Code Compilation & Artifact Creation

2.1 Build Tools Installation

PHP Installation:

```
# Install PHP 8.1+ with required extensions
# Extensions: php-cli, php-xml, php-mbstring, php-curl, php-zip, php-gd, php-
intl, php-bcmath
```

Composer Installation:

```
# Download and install Composer globally
php -r "copy('https://getcomposer.org/installer','composer-setup.php');"
php composer-setup.php --install-dir=/usr/local/bin --filename=composer
```

Node.js Installation:

- Version: Node.js 18.x LTS
- Includes npm for frontend dependency management
- Required for Nextcloud’s JavaScript asset compilation

2.2 Jenkinsfile Configuration

The Jenkinsfile was created to automate the build process with the following stages:

Pipeline Structure:

```
pipeline {
    agent any

    options {
        timeout(time: 60, unit: 'MINUTES')
        skipDefaultCheckout(true)
    }

    environment {
        GIT_SSL_NO_VERIFY = "true"
        NODE_SKIP_PLATFORM_CHECK = "true"
    }

    stages {
        // Detailed stages below
    }
}
```

Key Stages:

1. **Checkout Stage:**
   - Uses shallow clone for performance
   - Depth: 1 (latest commit only)
   - Timeout: 60 minutes
2. **Tool Verification Stage:**
   - Checks availability of Git, PHP, Composer, Node.js, npm
   - Displays version information for debugging
3. **Composer Dependencies:**
   - Installs PHP backend dependencies
   - Command: `composer install --no-interaction --prefer-dist --no-progress`

   - Skips if `composer.json` not found
4. **NPM Dependencies:**
   - Installs frontend JavaScript dependencies
   - Command: `npm install --legacy-peer-deps`
   - Flag handles peer dependency conflicts
5. **Frontend Build:**
   - Compiles JavaScript and CSS assets
   - Command: `npm run build`
   - Only runs if build script exists in package.json
6. **PHP Syntax Check:**
   - Lints all PHP files for syntax errors
   - Command: `php -l` on all `.php` files
   - Prevents deployment of syntactically invalid code

2.3 Artifact Creation

Build Output:

- Compiled JavaScript bundles in `/build/` or `/dist/`
- PHP dependencies in `/vendor/`
- Configuration files validated
- All artifacts prepared for testing stages

Artifact Archival:

- Jenkins archives build artifacts automatically
- Pattern: `build/**/*, dist/**/*, apps/**/build/**/*`
- Allows empty archives to prevent failures

---

3. Test Stage: Automated Testing

3.1 Backend Testing with PHPUnit

Test Structure:

```
backend_tests/
├── phpunit.xml              # PHPUnit configuration
└── tests/
    ├── Unit/               # Unit tests
    ├── Integration/        # Integration tests
    └── API/                # API endpoint tests
```

PHPUnit Configuration (phpunit.xml):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<phpunit bootstrap="vendor/autoload.php"
         colors="true"
         convertErrorsToExceptions="true"
         convertNoticesToExceptions="true"
         convertWarningsToExceptions="true">
    <testsuites>
        <testsuite name="Backend Test Suite">
            <directory>tests</directory>
        </testsuite>
    </testsuites>
</phpunit>
```

Test Execution in Pipeline:

```
stage('Run PHPUnit Tests') {
    steps {
        powershell '''
            if (Test-Path "vendor/bin/phpunit.bat") {
                Write-Host "Running PHPUnit..."
                vendor/bin/phpunit.bat --configuration phpunit.xml.dist
            } else {
                Write-Host "PHPUnit not found"
            }
        '''
    }
}
```

Backend Test Coverage:

- Authentication and authorization logic
- Database interaction layer
- API endpoint functionality
- Business logic validation
- Error handling mechanisms

3.2 Frontend Testing with Cypress

Test Structure:

```
frontend_tests/
├── cypress.config.js       # Cypress configuration
└── cypress/
    ├── e2e/               # End-to-end tests
    ├── fixtures/          # Test data
    └── support/           # Helper functions
```

Cypress Configuration (cypress.config.js):

```javascript
const { defineConfig } = require('cypress')
```

```
module.exports = defineConfig({
  e2e: {
    baseUrl: 'http://localhost:3000',
    specPattern: 'cypress/e2e/**/*.cy.js',
    supportFile: 'cypress/support/e2e.js',
    video: true,
    screenshotOnRunFailure: true,
  },
})
```

Test Execution in Pipeline:

```
stage('Frontend Tests (Cypress)') {
    steps {
        powershell '''
            cd frontend_tests
            npm install
            npx cypress run
        '''
    }
}
```

Frontend Test Coverage:

- User login and authentication flows
- Navigation and routing
- Form submissions and validation
- File upload/download functionality
- Error message display
- Responsive design elements

3.3 Test Reporting

JUnit XML Integration:

- PHPUnit generates JUnit-compatible XML reports
- Cypress configured to output test results in JUnit format
- Jenkins displays test results in pipeline view

Test Result Collection:

```
post {
    always {
        junit allowEmptyResults: true, testResults: '**/junit.xml'
    }
}
```

---

4. Staging Stage: AWS Deployment

4.1 AWS Infrastructure Setup

EC2 Instance Configuration:

- **AMI:** Ubuntu 22.04 LTS
- **Instance Type:** t2.medium or t3.medium
- **Storage:** 30-50 GB
- **Security Group Rules:**
  - HTTP (80): 0.0.0.0/0
  - HTTPS (443): 0.0.0.0/0
  - SSH (22): Restricted to admin IP
- **IAM Role:** EC2 role with CodeDeploy permissions

EC2 Instance Preparation:

```
# Update system packages
sudo apt update
sudo apt upgrade -y

# Install PHP and extensions
sudo apt install -y php php-cli php-fpm php-xml php-mbstring \
    php-zip php-gd php-curl php-bcmath php-intl php-mysql unzip git

# Install Nginx web server
sudo apt install -y nginx
sudo systemctl enable nginx
sudo systemctl start nginx

# Configure web root
sudo mkdir -p /var/www/nextcloud-staging
sudo chown -R www-data:www-data /var/www/nextcloud-staging
```

4.2 AWS CodeDeploy Agent Installation

```
# Install dependencies
sudo apt install -y ruby-full wget

# Download CodeDeploy agent installer
cd /home/ubuntu
wget https://aws-codedeploy-{region}.s3.amazonaws.com/latest/install

# Install and start agent
sudo chmod +x ./install
sudo ./install auto
sudo systemctl start codedeploy-agent
sudo systemctl enable codedeploy-agent
```

```
# Verify agent status
sudo systemctl status codedeploy-agent
```

4.3 S3 Bucket Configuration

Bucket Creation:

- Name: `nextcloud-staging-artifacts-{unique-id}`
- Region: Same as EC2 instance
- Block public access: Enabled
- Versioning: Enabled (recommended)
- Encryption: AES-256

Bucket Policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "codedeploy.amazonaws.com"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::nextcloud-staging-artifacts-{id}/*"
    }
  ]
}
```

4.4 CodeDeploy Application Setup

Application Creation:

1. AWS Console → CodeDeploy → Applications → Create Application
2. Application name: `Nextcloud-Staging`
3. Compute platform: EC2/On-premises

Deployment Group Configuration:

1. Name: `StagingGroup`
2. Service role: Create IAM role with CodeDeploy permissions
3. Deployment type: In-place
4. Environment configuration:
   - EC2 instances
   - Tag filter: Name = `nextcloud-staging`
5. Deployment settings: CodeDeployDefault.AllAtOnce
6. Load balancer: Disabled (for staging)

4.5 Deployment Specification (appspec.yml)

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/nextcloud-staging/

hooks:
  BeforeInstall:
    - location: scripts/stop_server.sh
      timeout: 60
      runas: ubuntu

  AfterInstall:
    - location: scripts/install_dependencies.sh
      timeout: 200
      runas: ubuntu

  ApplicationStart:
    - location: scripts/start_server.sh
      timeout: 60
      runas: ubuntu
```

4.6 Deployment Scripts

scripts/stop_server.sh:

```
#!/bin/bash
sudo systemctl stop nginx || true
```

scripts/install_dependencies.sh:

```
#!/bin/bash
cd /var/www/nextcloud-staging/

# Install Composer dependencies
if [ -f composer.json ]; then
    composer install --no-interaction --prefer-dist
fi

# Set proper permissions
sudo chown -R www-data:www-data /var/www/nextcloud-staging/
sudo chmod -R 755 /var/www/nextcloud-staging/

# Make scripts executable
chmod +x /var/www/nextcloud-staging/scripts/*.sh
```

scripts/start_server.sh:

```
#!/bin/bash
sudo systemctl restart php*-fpm || true
sudo systemctl restart nginx
```

4.7 Jenkins AWS Integration

AWS Credentials Configuration:

1. Manage Jenkins → Manage Credentials → Global → Add Credentials
2. Kind: AWS Credentials
3. ID: `aws-creds`
4. Access Key ID: From IAM user
5. Secret Access Key: From IAM user

Deployment Pipeline Stages:

```
stage('Package Artifact') {
    steps {
        powershell '''
            Compress-Archive -Path * -DestinationPath build.zip -Force
        '''
    }
}

stage('Upload to S3') {
    steps {
        powershell '''
            aws s3 cp build.zip s3://nextcloud-staging-artifacts/build-
${BUILD_NUMBER}.zip
        '''
    }
}

stage('Trigger CodeDeploy') {
    steps {
        powershell '''
            aws deploy create-deployment `
                --application-name Nextcloud-Staging `
                --deployment-group-name StagingGroup `
                --s3-location bucket=nextcloud-staging-
artifacts,bundleType=zip,key=build-${BUILD_NUMBER}.zip
        '''
    }
}
```

4.8 Deployment Validation

Automated Validation:

- Health check endpoint verification

- HTTP response code validation (200 OK)
- Critical functionality smoke tests

Manual Validation Process:

1. SSH into EC2 instance
2. Verify file deployment: `ls -la /var/www/nextcloud-staging/`
3. Check Nginx logs: `sudo tail -f /var/log/nginx/error.log`
4. Access application via browser: `http://{EC2-PUBLIC-IP}/`
5. Perform exploratory testing of key features

---

5. Key Challenges and Solutions

5.1 Large Repository Size

Challenge:

- Nextcloud server repository contains 1.1M+ objects
- Git clone operations failing with RPC errors
- Windows Git client limitations

Solution:

- Implemented shallow clone with depth=1
- Increased Git HTTP buffer: `http.postBuffer=524288000`
- Configured request buffer: `http.maxRequestBuffer=1000m`
- Disabled compression during clone: `core.compression=0`

5.2 Windows-Specific Issues

Challenge:

- Shell script execution not available on Windows
- Path length limitations
- Line ending differences (CRLF vs LF)

Solution:

- Used PowerShell for all pipeline commands
- Configured Git to handle line endings: `core.autocrlf=input`
- Deployment scripts execute on Linux EC2, not Windows

5.3 Network Connectivity

Challenge:

- Slow download speeds affecting Git operations
- Connection timeouts during large transfers

Solution:

- Extended Git timeout settings
- Configured low speed limits to prevent premature failures
- Used regional AWS services to minimize latency

---

6. Pipeline Execution Flow

6.1 Complete Pipeline Flow

```
1. Developer pushes code to GitHub
        ↓
2. GitHub webhook triggers Jenkins
        ↓
3. Jenkins performs shallow clone
        ↓
4. Build stage: Install dependencies & compile
        ↓
5. Backend tests: PHPUnit executes test suite
        ↓
6. Frontend tests: Cypress runs UI tests
        ↓
7. Package: Create deployment ZIP artifact
        ↓
8. Upload: Push artifact to S3 bucket
        ↓
9. Deploy: CodeDeploy pulls artifact and deploys to EC2
        ↓
10. Validation: Automated health checks + manual testing
```

6.2 Pipeline Execution Time

- Checkout: ~5-10 minutes (shallow clone)
- Build: ~3-5 minutes
- Backend Tests: ~2-3 minutes
- Frontend Tests: ~5-7 minutes
- Packaging & Upload: ~1-2 minutes
- Deployment: ~3-5 minutes
- **Total Average:** 19-32 minutes per deployment

---

7. Monitoring and Logging

7.1 Jenkins Pipeline Logs

Log Access:

- Pipeline view: Shows stage-by-stage execution
- Console output: Detailed command execution logs

- Blue Ocean: Visual pipeline representation

Key Metrics Tracked:

- Build success/failure rate
- Test pass/fail counts
- Stage execution duration
- Artifact size

7.2 AWS CloudWatch Integration

CodeDeploy Logs:

- Deployment lifecycle events
- Script execution output
- Error messages and stack traces

EC2 System Logs:

- Application logs: `/var/log/nextcloud/`
- Nginx access logs: `/var/log/nginx/access.log`
- Nginx error logs: `/var/log/nginx/error.log`
- PHP-FPM logs: `/var/log/php*-fpm.log`

7.3 Application Health Monitoring

Health Check Endpoints:

- Basic availability: `http://{ec2-ip}/status.php`
- Database connectivity check
- File system access verification

Monitoring Commands:

```
# Check Nginx status
sudo systemctl status nginx

# Check PHP-FPM status
sudo systemctl status php*-fpm

# Check CodeDeploy agent
sudo systemctl status codedeploy-agent

# View recent deployments
sudo tail -50 /var/log/aws/codedeploy-agent/codedeploy-agent.log
```

---

8. Best Practices Implemented

8.1 Version Control

- Feature branch workflow (main ← develop ← feature/*)
- Protected branches with required reviews
- Descriptive commit messages
- Semantic versioning for releases

8.2 Testing Strategy

- Unit tests for isolated component testing
- Integration tests for service interaction
- End-to-end tests for user workflows
- Automated test execution on every commit

8.3 Security

- Secrets stored in Jenkins credentials manager
- IAM roles with least privilege principle
- Security groups with minimal necessary access
- No hardcoded credentials in code or scripts

8.4 Deployment Strategy

- Separate staging environment for pre-production testing
- Automated rollback capability via CodeDeploy
- Zero-downtime deployment using health checks
- Incremental deployment validation

---

9. Tools and Technologies Summary

Category	Tool/Technology	Purpose
Version Control	GitHub	Source code repository and collaboration
CI/CD Orchestration	Jenkins	Pipeline automation and job management
Build Tools	Composer, npm	Dependency management and asset compilation
Backend Testing	PHPUnit	PHP unit and integration testing
Frontend Testing	Cypress	End-to-end UI testing
Cloud Platform	AWS	Infrastructure hosting
Deployment	AWS CodeDeploy	Automated application deployment
Storage	AWS S3	Artifact storage
Compute	AWS EC2	Application hosting
Web Server	Nginx	HTTP server and reverse proxy

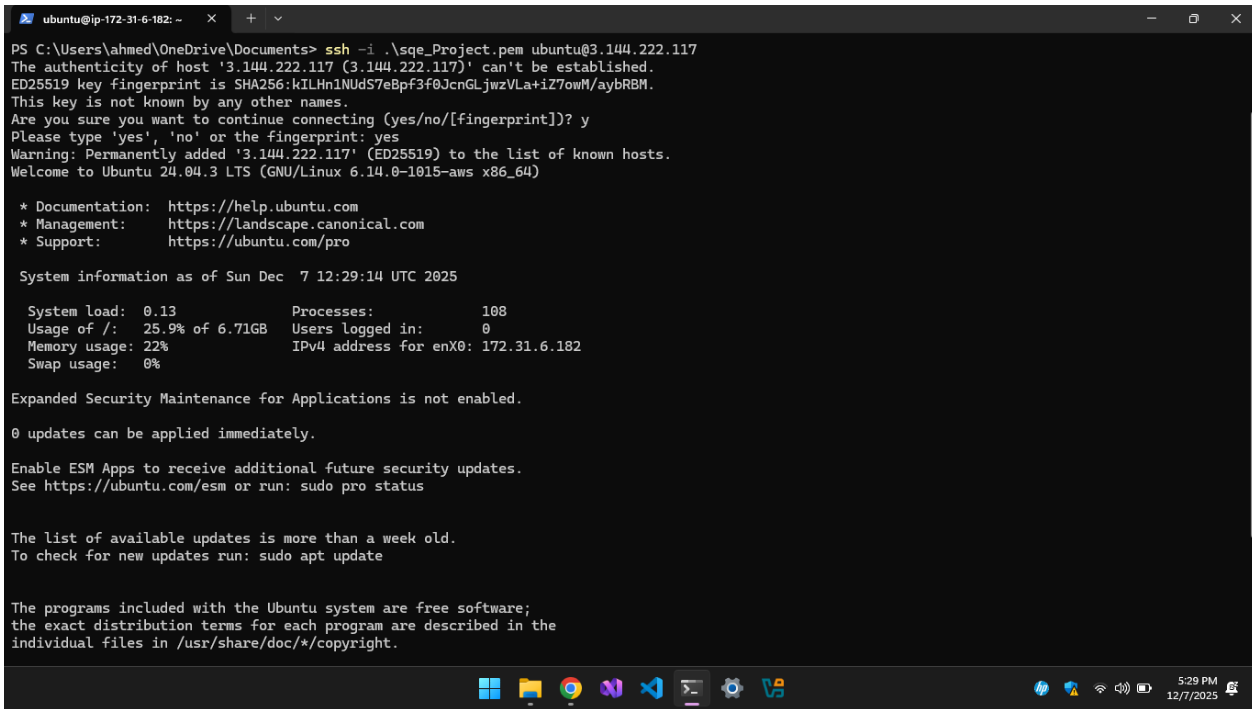
Category	Tool/Technology	Purpose
Runtime	PHP 8.1+, Node.js 18	Application execution environments

10. Conclusion

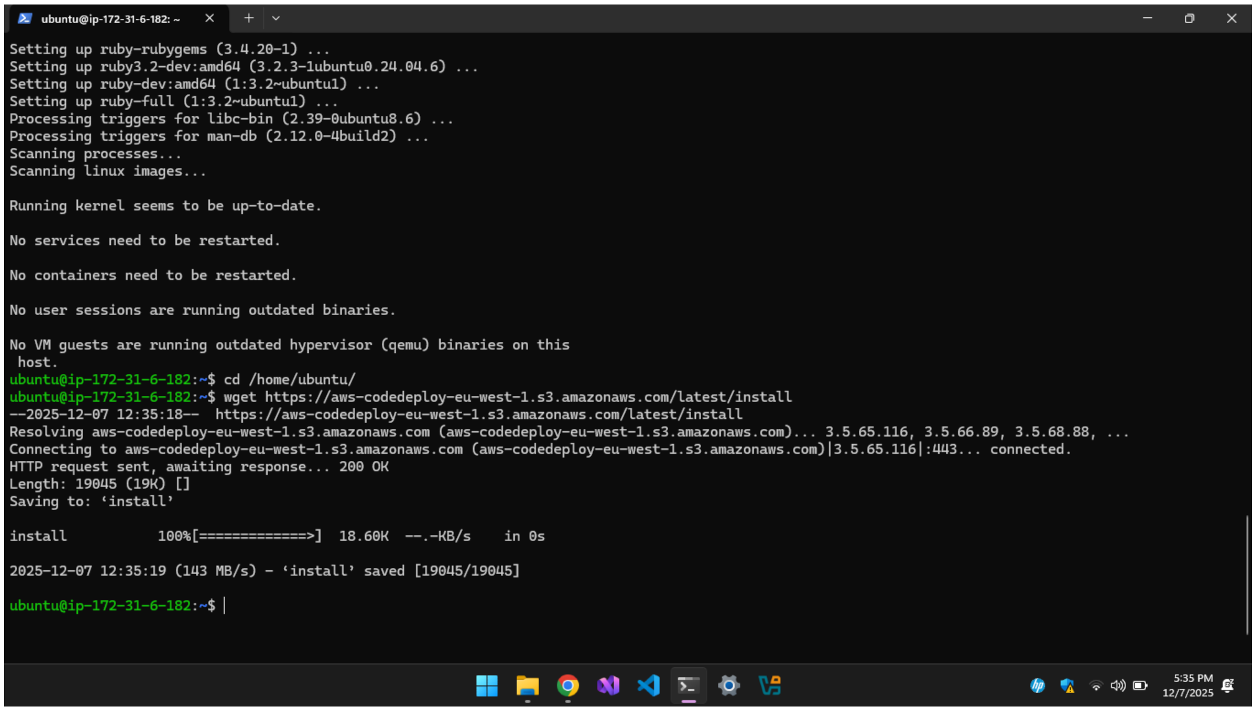
This project successfully implemented a comprehensive CI/CD pipeline for the Nextcloud Server application, demonstrating:

- **Automated Build Process:** Efficient compilation and dependency resolution
- **Comprehensive Testing:** Both backend (PHPUnit) and frontend (Cypress) test coverage
- **Reliable Deployment:** Automated staging deployment via AWS CodeDeploy
- **Quality Assurance:** Multiple validation gates ensuring code quality
- **Scalability:** Architecture supports future enhancements and production deployment

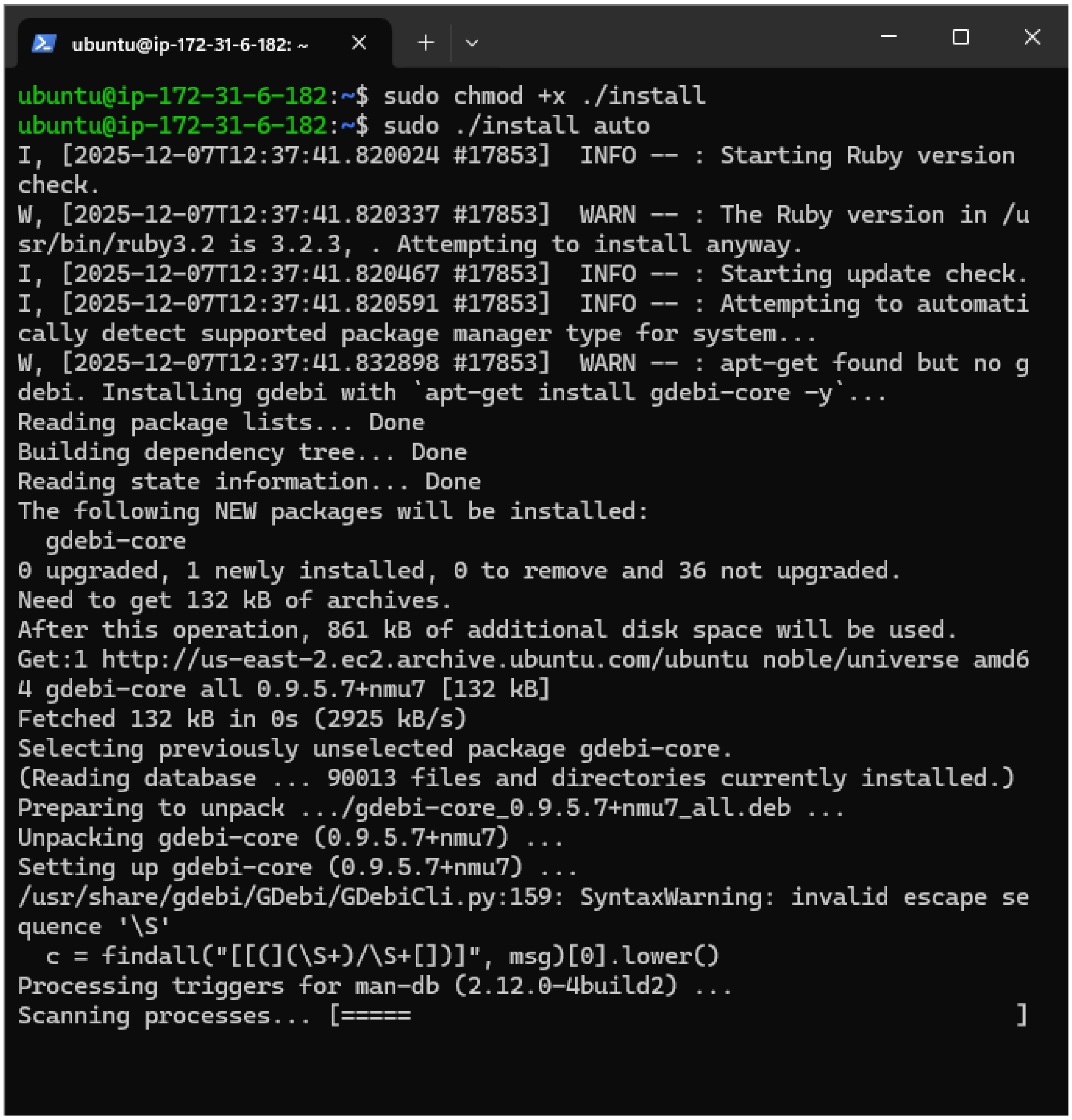
The pipeline provides a solid foundation for continuous integration and delivery, ensuring that code changes are automatically tested and deployed to staging environments with minimal manual intervention. The implementation follows industry best practices and can be extended to include production deployment, monitoring, and advanced deployment strategies.



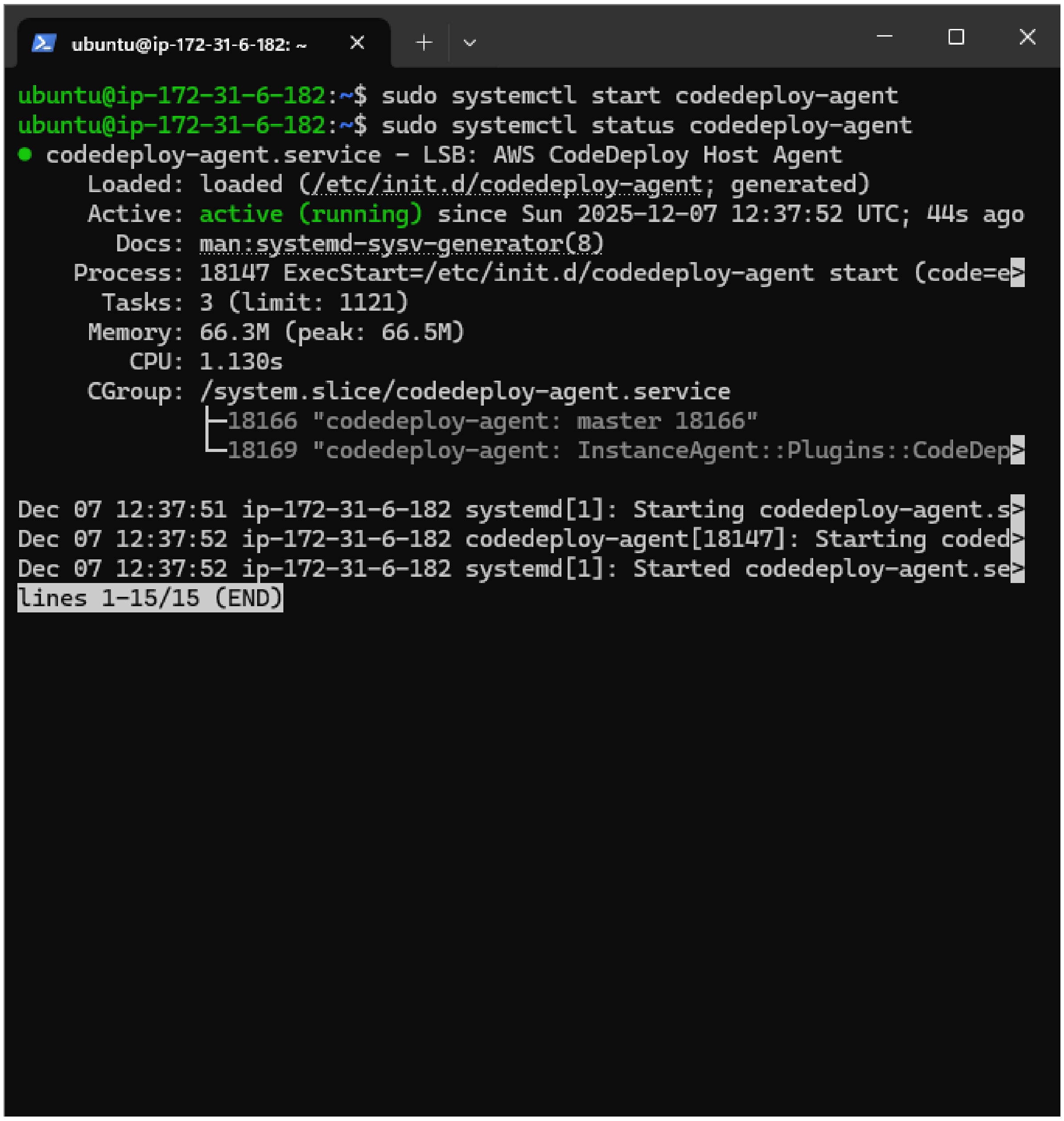
```
PS C:\Users\ahmed\OneDrive\Documents> ssh -i .\sqe_Project.pem ubuntu@3.144.222.117
The authenticity of host '3.144.222.117 (3.144.222.117)' can't be established.
ED25519 key fingerprint is SHA256:kILHn1NUdS7eBpf3f0JcnGLjwzVLa+iZ7owM/aybRBM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '3.144.222.117' (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 Sun Dec  7 12:29:14 UTC 2025

  System load:  0.13              Processes:             108
  Usage of /:   25.9% of 6.71GB   Users logged in:       0
  Memory usage: 22%               IPv4 address for enX0: 172.31.6.182
  Swap usage:   0%

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


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

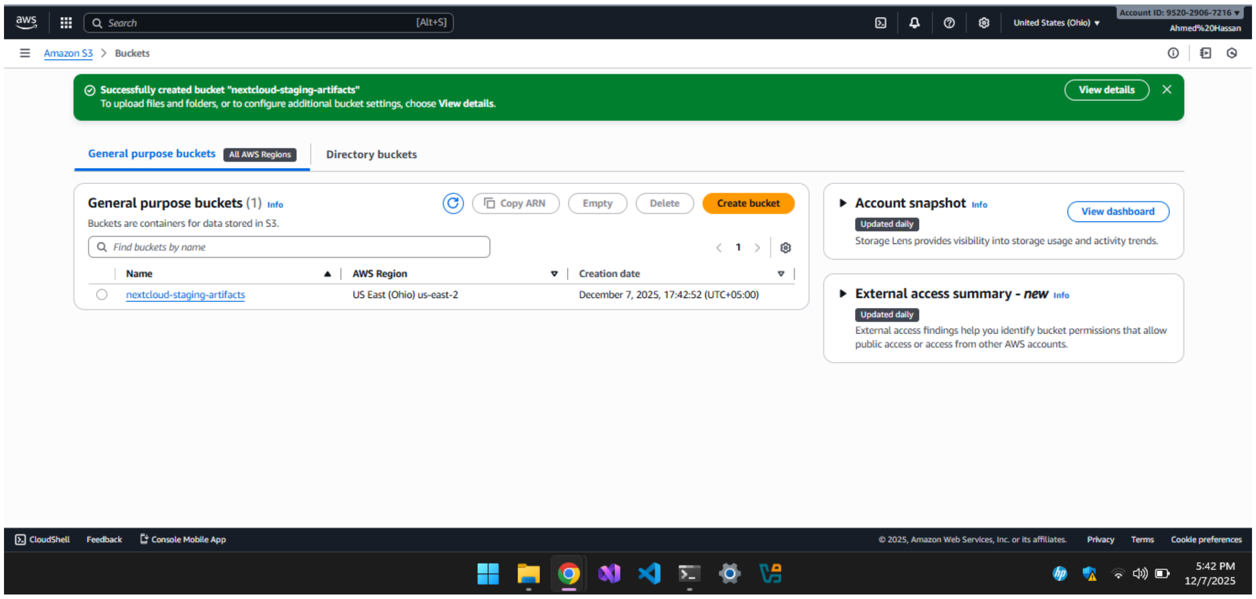


```
Setting up ruby-rubygems (3.4.20-1) ...
Setting up ruby3.2-dev:amd64 (3.2.3-1ubuntu0.24.04.6) ...
Setting up ruby-dev:amd64 (1:3.2~ubuntu1) ...
Setting up ruby-full (1:3.2~ubuntu1) ...
Processing triggers for libc-bin (2.39-0ubuntu8.6) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this
 host.
ubuntu@ip-172-31-6-182:~$ cd /home/ubuntu/
ubuntu@ip-172-31-6-182:~$ wget https://aws-codedeploy-eu-west-1.s3.amazonaws.com/latest/install
--2025-12-07 12:35:18--  https://aws-codedeploy-eu-west-1.s3.amazonaws.com/latest/install
Resolving aws-codedeploy-eu-west-1.s3.amazonaws.com (aws-codedeploy-eu-west-1.s3.amazonaws.com)... 3.5.65.116, 3.5.66.89, 3.5.68.88, ...
Connecting to aws-codedeploy-eu-west-1.s3.amazonaws.com (aws-codedeploy-eu-west-1.s3.amazonaws.com)|3.5.65.116|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19045 (19K) []
Saving to: ‘install’

install           100%[=============>]  18.60K  --.-KB/s    in 0s

2025-12-07 12:35:19 (143 MB/s) - ‘install’ saved [19045/19045]

ubuntu@ip-172-31-6-182:~$
```



```
ubuntu@ip-172-31-6-182:~$ sudo chmod +x ./install
ubuntu@ip-172-31-6-182:~$ sudo ./install auto
I, [2025-12-07T12:37:41.820024 #17853]  INFO -- : Starting Ruby version
check.
W, [2025-12-07T12:37:41.820337 #17853]  WARN -- : The Ruby version in /u
sr/bin/ruby3.2 is 3.2.3, . Attempting to install anyway.
I, [2025-12-07T12:37:41.820467 #17853]  INFO -- : Starting update check.
I, [2025-12-07T12:37:41.820591 #17853]  INFO -- : Attempting to automati
cally detect supported package manager type for system...
W, [2025-12-07T12:37:41.832898 #17853]  WARN -- : apt-get found but no g
debi. 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 36 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-2.ec2.archive.ubuntu.com/ubuntu noble/universe amd6
4 gdebi-core all 0.9.5.7+nmu7 [132 kB]
Fetched 132 kB in 0s (2925 kB/s)
Selecting previously unselected package gdebi-core.
(Reading database ... 90013 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 se
quence '\S'
  c = findall("[[(](\S+)/\S+[])]", msg)[0].lower()
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes... [=====                                           ]
```


```
ubuntu@ip-172-31-6-182: ~
ubuntu@ip-172-31-6-182:~$ sudo systemctl start codedeploy-agent
ubuntu@ip-172-31-6-182:~$ 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 Sun 2025-12-07 12:37:52 UTC; 44s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 18147 ExecStart=/etc/init.d/codedeploy-agent start (code=e>
      Tasks: 3 (limit: 1121)
     Memory: 66.3M (peak: 66.5M)
        CPU: 1.130s
     CGroup: /system.slice/codedeploy-agent.service
             ├─18166 "codedeploy-agent: master 18166"
             └─18169 "codedeploy-agent: InstanceAgent::Plugins::CodeDep>

Dec 07 12:37:51 ip-172-31-6-182 systemd[1]: Starting codedeploy-agent.s>
Dec 07 12:37:52 ip-172-31-6-182 codedeploy-agent[18147]: Starting coded>
Dec 07 12:37:52 ip-172-31-6-182 systemd[1]: Started codedeploy-agent.se>
lines 1-15/15 (END)
```



Amazon S3 > Buckets

Successfully created bucket "nextcloud-staging-artifacts"
To upload files and folders, or to configure additional bucket settings, choose **View details**.

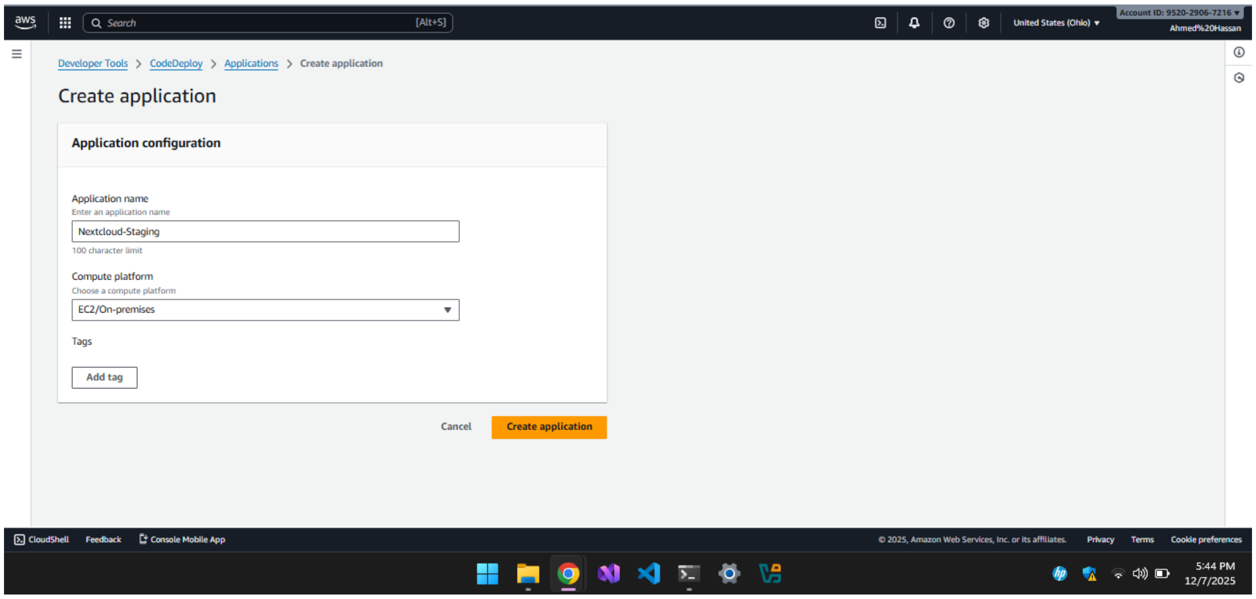
General purpose buckets (1) Info
Buckets are containers for data stored in S3.

Name	AWS Region	Creation date
nextcloud-staging-artifacts	US East (Ohio) us-east-2	December 7, 2025, 17:42:52 (UTC+05:00)

Account snapshot Info
Updated daily
Storage Lens provides visibility into storage usage and activity trends.

External access summary - *new* Info
Updated daily
External access findings help you identify bucket permissions that allow public access or access from other AWS accounts.

5:42 PM 12/7/2025



Developer Tools > CodeDeploy > Applications > Create application

Create application

Application configuration

Application name
Enter an application name
Nextcloud-Staging
100 character limit

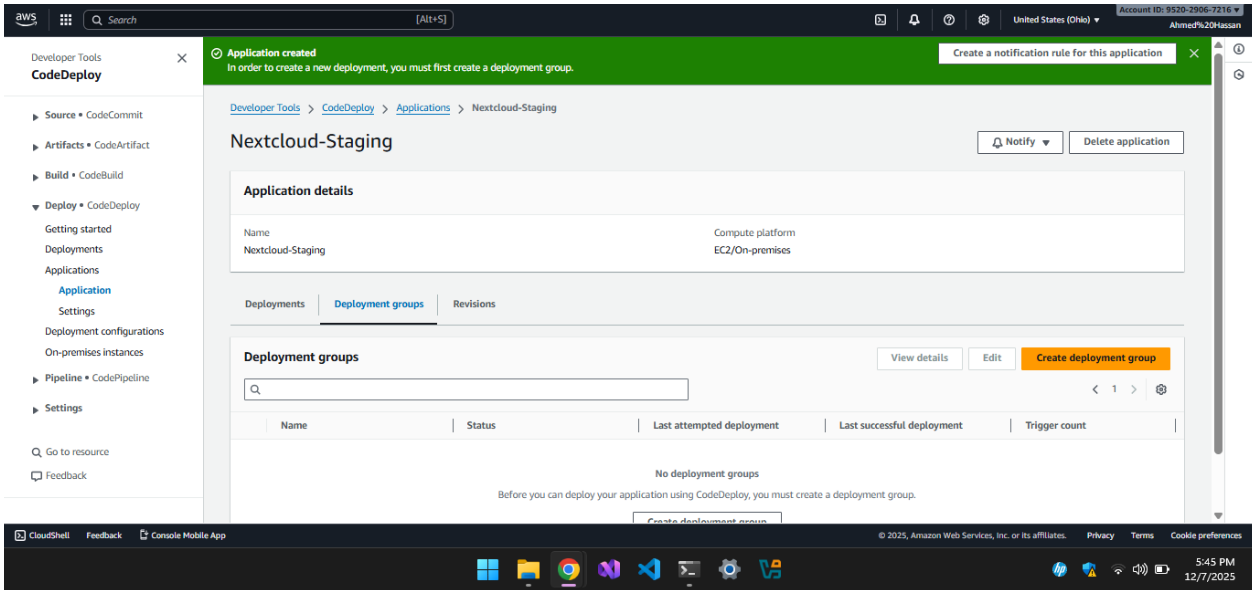
Compute platform
Choose a compute platform
EC2/On-premises

Tags

Add tag

Cancel Create application

5:44 PM 12/7/2025



Application created
In order to create a new deployment, you must first create a deployment group.

Developer Tools > CodeDeploy > Applications > Nextcloud-Staging

Nextcloud-Staging

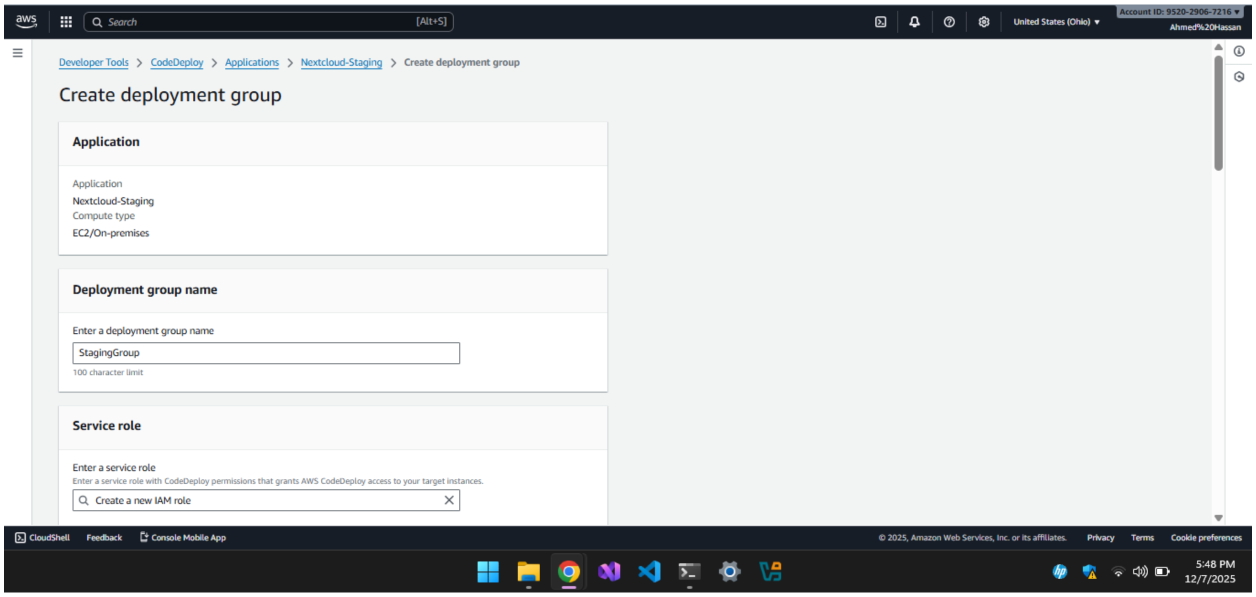
Application details

Name	Compute platform
Nextcloud-Staging	EC2/On-premises

Deployment groups

Name	Status	Last attempted deployment	Last successful deployment	Trigger count

No deployment groups
Before you can deploy your application using CodeDeploy, you must create a deployment group.



Developer Tools > CodeDeploy > Applications > Nextcloud-Staging > Create deployment group

Create deployment group

Application

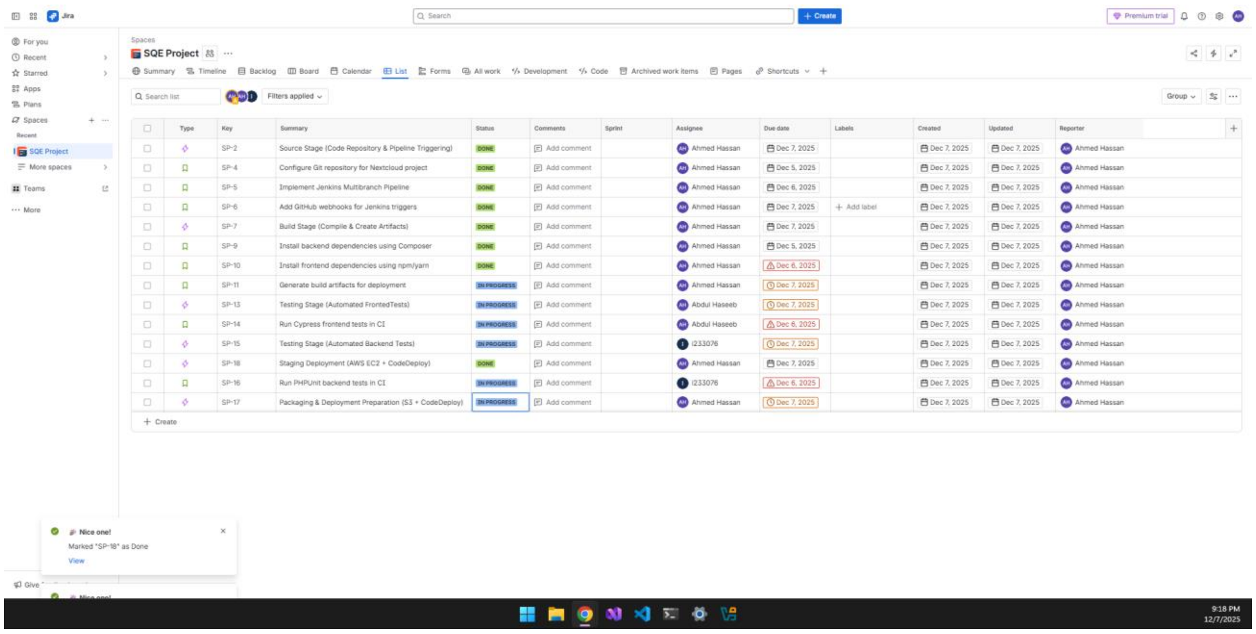
Application
Nextcloud-Staging
Compute type
EC2/On-premises

Deployment group name

Enter a deployment group name
StagingGroup
100 character limit

Service role

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



SQE Project

Type	Key	Summary	Status	Comments	Sprint	Assignee	Due date	Labels	Created	Updated	Reporter
⚡	SP-2	Source Stage (Code Repository & Pipeline Triggering)	DONE	Add comment		Ahmed Hassan	Dec 7, 2025		Dec 7, 2025	Dec 7, 2025	Ahmed Hassan
🔖	SP-4	Configure Git repository for Nextcloud project	DONE	Add comment		Ahmed Hassan	Dec 5, 2025		Dec 7, 2025	Dec 7, 2025	Ahmed Hassan
🔖	SP-5	Implement Jenkins Multibranch Pipeline	DONE	Add comment		Ahmed Hassan	Dec 6, 2025		Dec 7, 2025	Dec 7, 2025	Ahmed Hassan
🔖	SP-6	Add GitHub webhooks for Jenkins triggers	DONE	Add comment		Ahmed Hassan	Dec 7, 2025	+ Add label	Dec 7, 2025	Dec 7, 2025	Ahmed Hassan
⚡	SP-7	Build Stage (Compile & Create Artifacts)	DONE	Add comment		Ahmed Hassan	Dec 7, 2025		Dec 7, 2025	Dec 7, 2025	Ahmed Hassan
🔖	SP-9	Install backend dependencies using Composer	DONE	Add comment		Ahmed Hassan	Dec 5, 2025		Dec 7, 2025	Dec 7, 2025	Ahmed Hassan
🔖	SP-10	Install frontend dependencies using npm/yarn	DONE	Add comment		Ahmed Hassan	⚠ Dec 6, 2025		Dec 7, 2025	Dec 7, 2025	Ahmed Hassan
🔖	SP-11	Generate build artifacts for deployment	IN PROGRESS	Add comment		Ahmed Hassan	Dec 7, 2025		Dec 7, 2025	Dec 7, 2025	Ahmed Hassan
⚡	SP-13	Testing Stage (Automated FrontedTests)	IN PROGRESS	Add comment		Abdul Haseeb	Dec 7, 2025		Dec 7, 2025	Dec 7, 2025	Ahmed Hassan
🔖	SP-14	Run Cypress frontend tests in CI	IN PROGRESS	Add comment		Abdul Haseeb	⚠ Dec 6, 2025		Dec 7, 2025	Dec 7, 2025	Ahmed Hassan
⚡	SP-15	Testing Stage (Automated Backend Tests)	IN PROGRESS	Add comment		i233076	Dec 7, 2025		Dec 7, 2025	Dec 7, 2025	Ahmed Hassan
⚡	SP-18	Staging Deployment (AWS EC2 + CodeDeploy)	DONE	Add comment		Ahmed Hassan	Dec 7, 2025		Dec 7, 2025	Dec 7, 2025	Ahmed Hassan
🔖	SP-16	Run PHPUnit backend tests in CI	IN PROGRESS	Add comment		i233076	⚠ Dec 6, 2025		Dec 7, 2025	Dec 7, 2025	Ahmed Hassan
⚡	SP-17	Packaging & Deployment Preparation (S3 + CodeDeploy)	IN PROGRESS	Add comment		Ahmed Hassan	Dec 7, 2025		Dec 7, 2025	Dec 7, 2025	Ahmed Hassan

Nice one!
Marked "SP-18" as Done

Project Completed By: Ahmed Hassan, Abdul Haseeb, Saad Ahmed

Date: December 2025

Course: Software Quality Engineering