

AWS DevOps

AWS CodeDeploy

참고링크 : https://docs.aws.amazon.com/ko_kr/codedeploy/latest/userguide/instances-ec2-create.html

온프레미스 서버인 경우 아래 링크 참조

https://docs.aws.amazon.com/ko_kr/codedeploy/latest/userguide/instances-on-premises.html

참고 링크 : https://docs.aws.amazon.com/ko_kr/codedeploy/latest/userguide/instances-ec2-create.html

EC2 생성 CLI 명령어

// VPC 생성

\$ aws ec2 create-vpc --cidr-block 10.0.0.0/16

// DNS 활성화

\$ aws ec2 modify-vpc-attribute --vpc-id vpc-07f21f8ccee1876d2 --enable-dns-hostnames

// VPC 내 Subnet 생성

\$ aws ec2 create-subnet --vpc-id vpc-07f21f8ccee1876d2 --availability-zone us-east-1a --cidr-block 10.0.0.0/24

// VPC 내 RouteTable 생성

\$ aws ec2 create-route-table --vpc-id vpc-07f21f8ccee1876d2

// RouteTable 에 Subnet 포함시키기

\$ aws ec2 associate-route-table --route-table-id rtb-0213dbaf47d5614e2 --subnet-id subnet-0292280b7b7f506c1

// Internet Gateway 생성

\$ aws ec2 create-internet-gateway

// VPC 에 Internet Gateway 결합

\$ aws ec2 attach-internet-gateway --internet-gateway-id igw-021d9f9d185a44a93 --vpc-id vpc-07f21f8ccee1876d2

// RouteTable 에 Internet Gateway 포함

\$ aws ec2 create-route --route-table-id rtb-0213dbaf47d5614e2 --destination-cidr-block 0.0.0.0/0 --gateway-id igw-021d9f9d185a44a93

```
// EC2 에 사용할 SSH Key 생성
$ aws ec2 create-key-pair --key-name key0622 --query 'KeyMaterial' --output text>
key0622_keypair.pem

// Key 값 확인
$ cat key0622_keypair.pem

// Security Group 생성
$ aws ec2 create-security-group --group-name sg01 --description "for my ec2" --
vpc-id vpc-07f21f8ccee1876d2

// 현재 나의 IP 체크 후 SSH 로 접근시 나의 IP 만 허용
$ curl http://checkip.amazonaws.com
$ aws ec2 authorize-security-group-ingress --group-id sg-035ccfb67e8afe07b --
protocol tcp --port 22 --cidr 1.235.44.150/32

// 80 포트 열기
$ aws ec2 authorize-security-group-ingress --group-id sg-035ccfb67e8afe07b --
protocol tcp --port 80 --cidr 0.0.0.0/0

// EC2 생성
$ aws ec2 run-instances --image-id ami-085925f297f89fce1 --count 1 --instance-
type t2.micro --key-name key0622 --security-group-ids sg-035ccfb67e8afe07b --
subnet-id subnet-0292280b7b7f506c1

// Elastic IP 생성 후 EC 에 연결
$ aws ec2 allocate-address
$ aws ec2 associate-address --instance-id i-074d916e23c798824 --allocation-id
eipalloc-06d01a6e2ef78f5cb

// Key 권한 설정 후, Ubuntu 에 연결
$ chmod 0600 key0622_keypair.pem
$ ssh -i key0622_keypair.pem ubuntu@18.233.252.89
```

IAM Role 생성

1. **Create Role** 에서 **EC2** 선택 - **EC2** 에서 **S3** 에 접근할 수 있도록 한다.

Create role

1 2 3 4

▼ Attach permissions policies

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

Create policy



Filter policies ▼

Q S3

Showing 4 results

	Policy name ▼	Used as
<input type="checkbox"/>	▶ AmazonDMSRedshiftS3Role	None
<input type="checkbox"/>	▶ AmazonS3FullAccess	None
<input checked="" type="checkbox"/>	▶ AmazonS3ReadOnlyAccess	Permissions policy (1)

2. **Create Role** 에서 **CodeDeploy** 선택.

CodeDeploy

Select your use case

CodeDeploy

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

3. EC2 생성 시, 아래 스샷의 User data 에 다음과 같은 코드를 넣어준다. Ubuntu 와 Amazon Linux 다름. -> Agent 설치.
4. 버킷 이름을 지역에 따라 바꾼다.
5. 예) **bucket-name**을 **aws-codedeploy-us-east-1**로 대체한다.

```
#!/bin/bash
apt -y update
apt -y install ruby
apt -y install wget
cd /home/ubuntu
wget https://bucket-name.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

Step 3: Configure Instance Details

Additional charges may apply when launching Dedicated instances.

Elastic Inference ⓘ

☐ Add an Elastic Inference accelerator

Additional charges apply.

T2/T3 Unlimited ⓘ

☐ Enable

Additional charges may apply

File systems ⓘ

Add file system



Create new file system

▼ Network interfaces ⓘ

Device	Network Interface	Subnet	Primary IP	Secondary IP addresses
eth0	New network interface ▼	subnet-05d0148f ▼	Auto-assign	Add IP

Add Device

▼ Advanced Details

Metadata accessible ⓘ

Enabled ▼

Metadata version ⓘ

V1 and V2 (token optional) ▼

Metadata token response hop limit ⓘ

1 ▼

User data ⓘ

☒ As text ☐ As file ☐ Input is already base64 encoded

(Optional)

// 그리고 EC2 에 태그도 정해 둔다.

Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with a key of `Environment` and a value of `Production`. A copy of a tag can be applied to volumes, instances or both.

Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key (128 characters maximum)	Value
------------------------------	-------

web

test

Add another tag

(Up to 50 tags maximum)

// EC2 에 접속하여, CodeDeploy 를 위한 Agent 가 설치되어 있는지 확인한다.

```
$ sudo service codedeploy-agent status
```

참고 : https://docs.aws.amazon.com/ko_kr/codedeploy/latest/userguide/codedeploy-agent-operations-verify.html

// 출력

- codedeploy-agent.service - LSB: AWS CodeDeploy Host Agent
Loaded: loaded (/etc/init.d/codedeploy-agent; generated)
Active: **inactive** (dead)
Docs: man:systemd-sysv-generator(8)

// 위와 같이 출력되면, AWS 의 CodeDeploy 서비스로 이동한다.

// 서비스를 시작하기 위해 아래 명령어 입력

```
$ sudo service codedeploy-agent start
```

// CodeDeploy Agent 의 버전 확인

```
$ cat /opt/codedeploy-agent/.version 또는 $ sudo dpkg -s codedeploy-agent
```

// Agent 의 재설치를 위해서 아래 페이지 참조

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

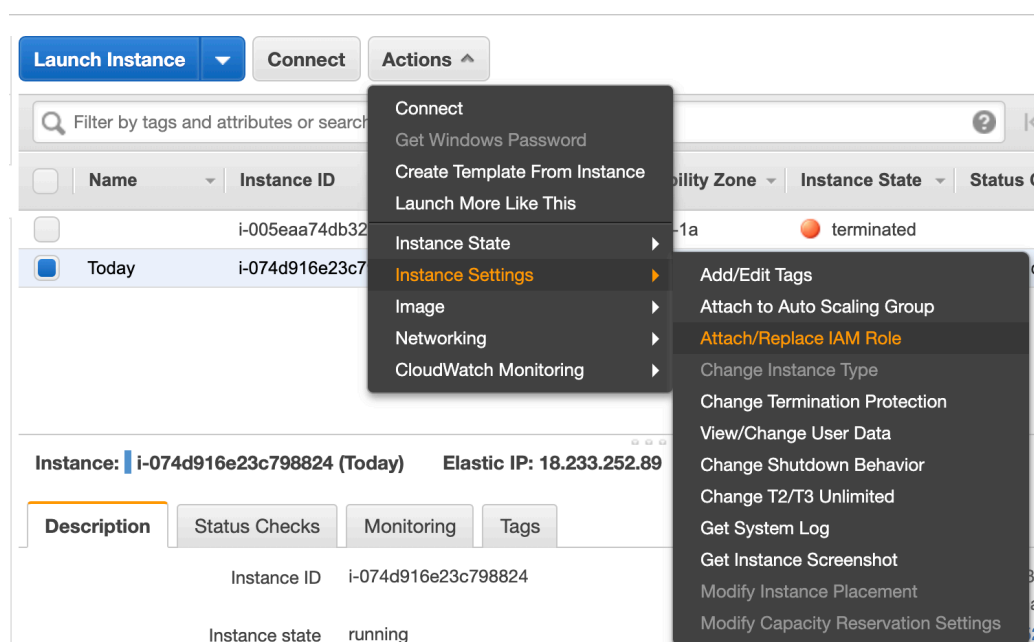
// Agent 의 업데이트

```
$ sudo /opt/codedeploy-agent/bin/install auto
```

// Agent 삭제

```
$ sudo dpkg --purge codedeploy-agent
```

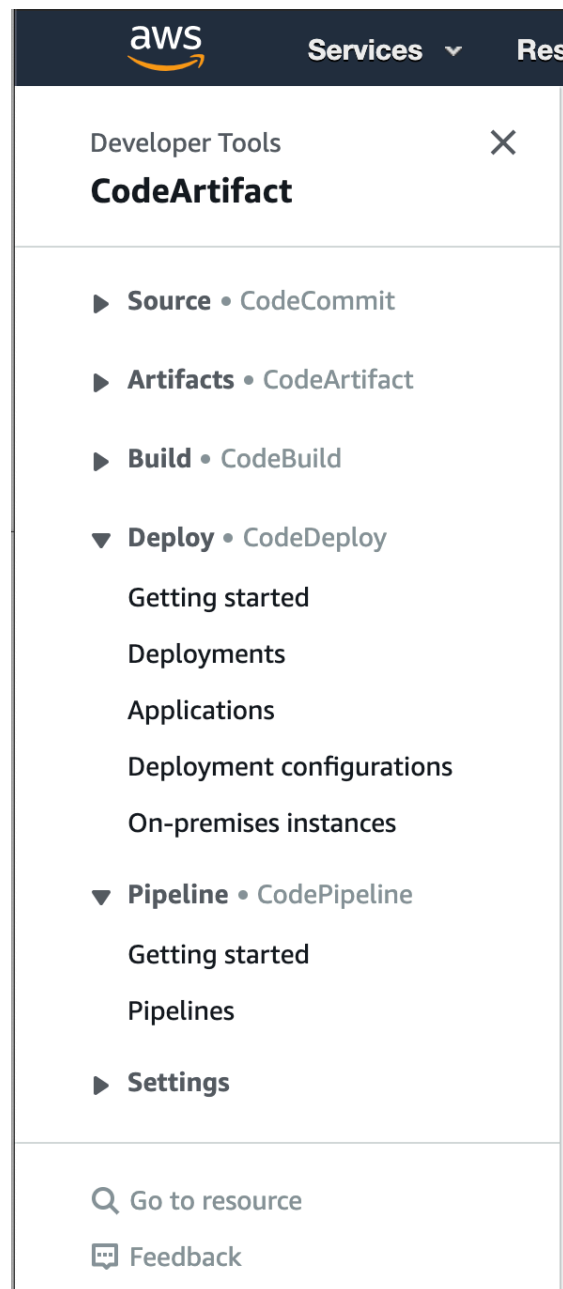
// EC2 에 롤 추가



CodeCommit, CodeArtifact, CodeBuild, CodeDeploy, CodePipeline

아래 스샷에서 위 명칭들의 메뉴가 보인다. 여기서 **CodeDeploy**, **CodePipeline** 에만 집중!

즉, **Deploy**, **Pipeline**



// 좌측 탭의 Application 메뉴로 이동하여 Create

Developer Tools > CodeDeploy > Applications > Create application

Create application

Application configuration

Application name
Enter an application name

100 character limit

Compute platform
Choose a compute platform

[Cancel](#) [Create application](#)

// Create Deployment Group 클릭

Developer Tools > CodeDeploy > Applications > FirstNodeOnEC2

FirstNodeOnEC2

[Notify](#) [Delete application](#)

Application details

Name	Compute platform
FirstNodeOnEC2	EC2/On-premises

[Deployments](#) [Deployment groups](#) [Revisions](#)

Deployment groups

[View details](#) [Edit](#) [Create deployment group](#)

< 1 > ⚙

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.				
Create deployment group				

Deployment group name

Enter a deployment group name

FirstGroup

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

arn:aws:iam::305163539117:role/codedeployRole



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 deployment group are registered with a load balancer, the old instances are deregistered

Environment configuration

Select any combination of Amazon EC2 Auto Scaling groups, Amazon EC2 instances, and on-premises instances 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](#)

Deployment settings

Deployment configuration

Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application is deployed and the success or failure conditions for a deployment.

CodeDeployDefault.AllAtOnce



or

[Create deployment configuration](#)

Load balancer

Select a load balancer to manage incoming traffic during the deployment process. The load balancer blocks traffic from each instance while it's being deployed to and allows traffic to it again after the deployment succeeds.

☐ Enable load balancing

// CodePipeline 메뉴로 이동 - Create pipeline 클릭

Developer Tools 
CodePipeline

- ▶ Source • CodeCommit
- ▶ Artifacts • CodeArtifact
- ▶ Build • CodeBuild
- ▶ Deploy • CodeDeploy
- ▶ Pipelines • CodePipeline

Developer Tools > CodePipeline > Pipelines

Pipelines Info



Notify ▼

[View history](#)

[Release change](#)

[Delete pipeline](#)

[Create pipeline](#)



1



Name

Most recent execution

Latest source revisions

// 기본 셋팅으로 생성

Pipeline settings

Pipeline name

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

No more than 100 characters

Service role

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

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

Role name

Type your service role name

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

▼ Advanced settings

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.


// GitHub 를 소스 프로바이더로 정하고 Connect 버튼을 눌러서 로그인 한다.


// Authorize 클릭




Authorize AWS CodePipeline (N. Virginia)

**AWS CodePipeline (N. Virginia)** by **aws-codesuite**
wants to access your **CodingSta** account

**Repository webhooks and services**
Admin access

**Repositories**
Public and private

Organization access

 krazure ✕

Grant

Authorize aws-codesuite

// Repository 와 Branch 선택

Add source stage [Info](#)

Source

Source provider

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

GitHub

Grant AWS CodePipeline access to your GitHub repository. This allows AWS CodePipeline to upload commits from GitHub to your pipeline.

Connected

✓ You have successfully configured the action with the provider.

Repository

🔍 CodingSta/EC2DevOps

Branch

🔍 master

Change detection options

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

☒ **GitHub webhooks (recommended)**
Use webhooks in GitHub to automatically start my pipeline when a change occurs

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

Cancel

Previous

Next

// Build Stage 는 Skip

Add build stage [Info](#)

Build - optional

Build provider

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

Cancel


Previous

Skip build stage

Next

// 아래와 같이 Deploy Stage 의 대상 선택.

Add deploy stage [Info](#)

 **You cannot skip this stage**
Pipelines must have at least two stages. Your second stage must be either a build or deployment stage. Choose a provider for either the build stage or deployment stage.

Deploy

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

AWS CodeDeploy ▼

Region

US East (N. Virginia) ▼

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

🔍 FirstNodeOnEC2 ✕

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

🔍 FirstGroup ✕

Cancel Previous **Next**

// Review 후 Create pipeline 클릭

Step 4: Add deploy stage

Deploy action provider

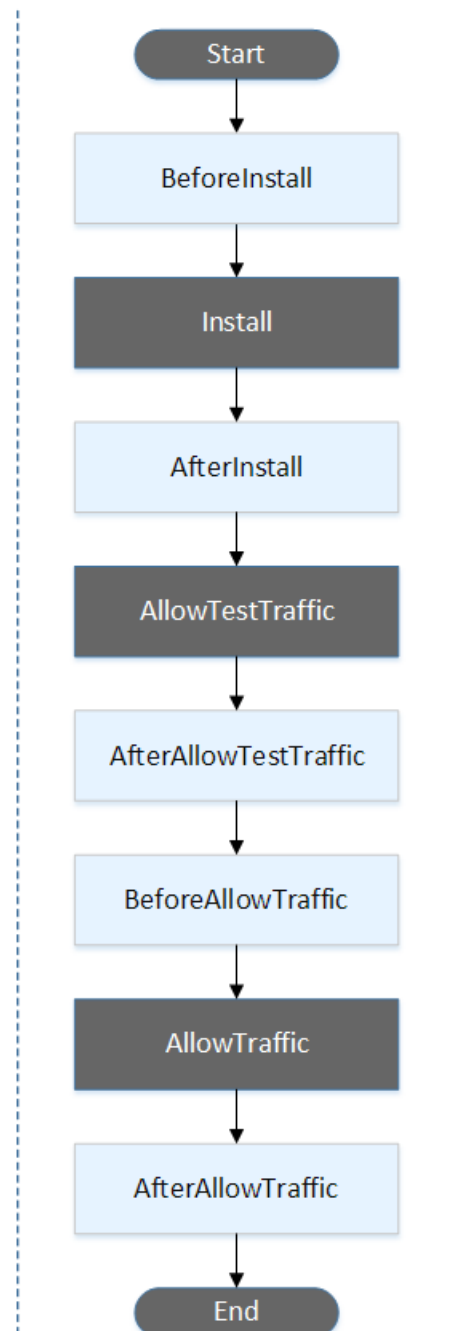
Deploy action provider
AWS CodeDeploy
ApplicationName
FirstNodeOnEC2
DeploymentGroupName
FirstGroup

Cancel Previous **Create pipeline**

Developer Tools

CodePipeline

- ▶ **Source** • CodeCommit
- ▶ **Artifacts** • CodeArtifact
- ▶ **Build** • CodeBuild
- ▶ **Deploy** • CodeDeploy
- ▼ **Pipeline** • CodePipeline
 - Getting started
 - Pipelines
 - Pipeline**
 - History
 - Settings
 - ▶ **Settings**



CodeDeploy FAQ 에 배포 라이프 사이클 이벤트의 설명을 전제합니다.

배포 라이프 사이클 이벤트	설명
ApplicationStop	배포의 첫 번째 라이프 사이클 이벤트이며, 수정 다운로드 전부터 시작합니다. 이 배포 라이프 사이클 이벤트에 사용되는 AppSpec 파일과 스크립트는 이전 성공적으로 배포 된 수정합니다. 응용 프로그램을 정상적으로 종료하거나 배포 준비로 최근에 설치 한 패키지를 제거하려면 ApplicationStop 배포 라이프 사이클 이벤트를 사용합니다.
DownloadBundle	이 배포 라이프 사이클 이벤트는 에이전트 인스턴스의 임시 위치에 개정 파일을 복사합니다. 이 배포 라이프 사이클 이벤트는 에이전트 용으로 예약되어 있으며, 사용자 스크립트의 실행에 사용할 수 없습니다.
BeforeInstall	BeforeInstall 라이프 사이클 이벤트는 파일의 암호화 및 현재 버전의 백업 작성 등 설치 작업에 사용합니다.
Install	이 배포 라이프 사이클 이벤트는 에이전트가 임시 위치에서 수정 파일을 최종 대상 폴더에 복사합니다. 이 배포 라이프 사이클 이벤트는 에이전트 용으로 예약되어 있으며, 사용자 스크립트의 실행에 사용할 수 없습니다.
AfterInstall	AfterInstall 배포 라이프 사이클 이벤트는 응용 프로그램의 설정 또는 파일의 권한 변경에 사용합니다.
ApplicationStart	기본적으로 이 배포의 라이프 사이클 이벤트는 ApplicationStop 에서 중지 된 서비스를 다시 시작하는 데 사용합니다.
ValidateService	ValidateService 마지막 배포 라이프 사이클 이벤트이며 배포가 성공적으로 완료되었는지 확인합니다.

EC2 에서 아래 폴더로 이동한다.

```
$ cd /opt/codedeploy-agent/deployment-root/deployment-instructions/
```

```
// 하위 파일을 보고 최근 성공한 파일 이름 확인.
```

```
065888e9-3885-400b-b021-af766edbe82f-cleanup
```

```
065888e9-3885-400b-b021-af766edbe82f-install.json
```

```
065888e9-3885-400b-b021-af766edbe82f_last_successful_install
```

```
065888e9-3885-400b-b021-af766edbe82f_most_recent_install
```

// 파일 내용 확인

```
$ cat /opt/codedeploy-agent/deployment-root/deployment-instructions/  
065888e9-3885-400b-b021-af766edbe82f_last_successful_install
```

// 출력

```
/opt/codedeploy-agent/deployment-root/065888e9-3885-400b-b021-af766edbe82f/d-  
VNENDRWW4
```

\$ sudo apt install tree

// Tree 명령어로 위 경로 하위 폴더 확인

```
$ tree /opt/codedeploy-agent/deployment-root/065888e9-3885-400b-b021-  
af766edbe82f/d-VNENDRWW4
```

// 아래 폴더로 이동하여 **scripts** 에서 **stop...** 삭제 및 **appspec.yml** 편집

```
ubuntu@ip-10-0-0-72:/opt/codedeploy-agent/deployment-root/deployment-instructions$ tree /op  
t-root/065888e9-3885-400b-b021-af766edbe82f/d-VNENDRWW4  
/opt/codedeploy-agent/deployment-root/065888e9-3885-400b-b021-af766edbe82f/d-VNENDRWW4  
├── bundle.tar  
├── deployment-archive  
│   ├── appspec.yml  
│   ├── index.js  
│   ├── package-lock.json  
│   ├── package.json  
│   └── scripts  
│       ├── install_dependencies.sh  
│       ├── start_server.sh  
│       ├── stop_server.sh  
│       └── validate_server.sh  
└── logs  
    └── scripts.log  
  
3 directories, 10 files
```

// 웹 콘솔에서 배포 할 때는 **--ignore-application-stop-failures** 옵션을 지정할 수 없다

// 첫 실행에서 start_server.sh 파일에서 **pm2 stop index.js** 을 주석 처리 한다.

// 첫 실행에서 install_dependencies.sh 파일에서 아래 부분을 주석 처리 한다.

cd /home/ubuntu/FirstNodeApp

sudo rm -rf node_modules