

1. Setup Cloud9 IDE, Launch Cloud9 IDE and write and run a simple Python program
2. Setup Cloud9 IDE, Launch Cloud9 IDE and write and run a simple HTML script

-> Search EC2 and check if all instances are not running -> Services (top left) -> Developer tools -> Cloud9 -> Click on create environment -> Enter the name -> Keep all setting on default -> Create

-> Search EC2 -> (There should be an Instance running) -> Search Cloud Formation -> Services (top left) -> Developer tools -> Cloud9 -> Click on open -> Create a new python file (hello.py) -> Print something -> Go to the previous page and delete environment

-> Search Cloud9 -> Go to running instance -> Terminate instance

3. Collaborate with other users from within cloud9 IDE.
4. Collaborate with other users and change access level for each user from within cloud9 IDE.

### Add Collaborators


-> Search IAM -> Left col (user) -> Create User -> Create Group (make sure don't recreate the password)

#### User details

User name

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ \_ - (hyphen)

☒ Provide user access to the AWS Management Console - *optional*  
If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

 Are you providing console access to a person?

User type

☐ Specify a user in Identity Center - Recommended  
We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.

☒ I want to create an IAM user  
We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keyspaces, or a backup credential for emergency account access.

Console password

☐ Autogenerated password  
You can view the password after you create the user.

☒ Custom password  
Enter a custom password for the user.

• Must be at least 8 characters long

• Must include at least three of the following mix of character types: uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), and symbols ! @ # \$ % ^ & \* ( ) \_ + = { } [ ] \ | ; ' " , . - (punctuation)

## Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

### Permissions options

☐ Add user to group

Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ Copy permissions

Copy all group memberships, attached managed policies, and inline policies from an existing user.

☒ Attach policies directly

Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

### Permissions policies (1/1133)

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



Create policy

cloud9

Filter by Type

All types

5 matches

< 1 > ⚙

<input type="checkbox"/>	Policy name	Type	Attached entities
<input type="checkbox"/>	<a href="#">AWSCloud9Administrator</a>	AWS managed	0
<input checked="" type="checkbox"/>	<a href="#">AWSCloud9EnvironmentMember</a>	AWS managed	1
<input type="checkbox"/>	<a href="#">AWSCloud9ServiceRolePolicy</a>	AWS managed	1
<input type="checkbox"/>	<a href="#">AWSCloud9SSMInstanceProfile</a>	AWS managed	1

## Create group

## Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

### Permissions options

☒ Add user to group

Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ Copy permissions

Copy all group memberships, attached managed policies, and inline policies from an existing user.

☐ Attach policies directly

Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

### User groups (1)



Create group

Search

< 1 > ⚙

<input type="checkbox"/>	Group name	Users	Attached policies	Created
<input type="checkbox"/>	<a href="#">group1</a>	0	<a href="#">AWSCloud9EnvironmentMember</a>	2023-10-25 (Now)

->Create user -> Return to users list ->Cloud 9 -> open

```

nec2-user:~/environment $ npm --v
10.2.0
nec2-user:~/environment $ npm --v
10.2.0
ec2-user:~/environment $ git --version
git version 2.40.1
ec2-user:~/environment $ node --version
v18.17.1
ec2-user:~/environment $ python --version
Python 3.8.16
ec2-user:~/environment $ 

```

How to colab?

Click on File-> Create new template file -> Create a python file -> Edit and save

Click on share on top right -> share -> type username and add user -> Copy the environment link and paste it in incognito

- 5. Build an Application using AWS CodePipeline, deploy Sample Application on EC2 instance.**
- 6. Build an Application using AWS CodePipeline, deploy Sample Application on EC2 instance, make changes to application code and deploy.**

in S3 bucket -> create bucket  
give name and create

go to iam user  
create role -> ec2 -> permission  
awselasticbeanstalkwebtier

next give role name  
and create role

open beanstalk-> environment  
create environment  
give name  
platform -> php  
next  
use existing service role  
ec2 instance select role

next-next-create  
after environment created

now go to code pipeline  
create pipeline  
github(version2)  
connect to github

select repository  
branch name-> master

skip build stage  
deploy provider->elastic beanstalk

give environment name  
next....  
create pipeline

now in deploy->domain->done

## **7. Install Terraform on Windows machine. Build, apply and destroy AWS EC2 using Terraform.**

**8. Test TypeScript code using SonarQube.**

**9. Test java code using SonarQube.**

**10. Test python code using SonarQube.**

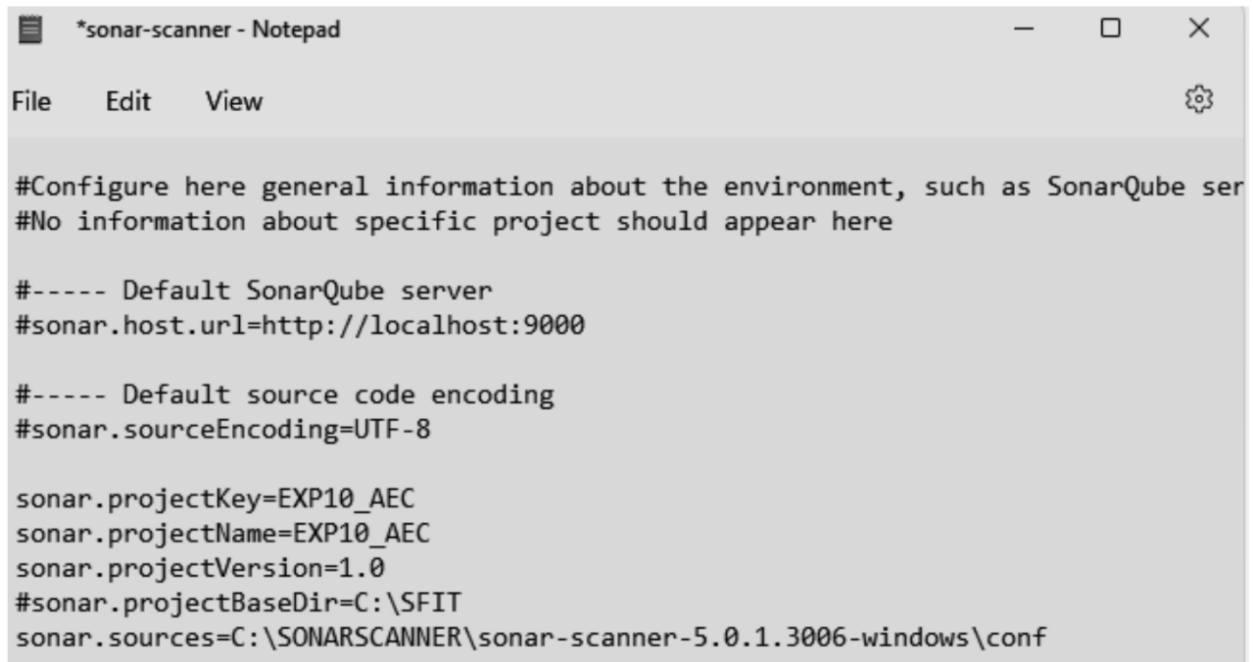
-> Go to <https://www.sonarsource.com/products/sonarqube/downloads/> -> Download community edition -> Extract -> Go to bin -> Copy path -> Set the environment variable

-> Go inside windows x64 dir, right click on start sonar -> Run as administrator -> Enter Project display name and project key -> branch main -> Next

-> Set up project for Clean as you Code -> Use the global setting (Radio) ->

-> 1 Provide a token -> Generate project token -> (Token name) Analyze "Exp10\_AEC" -> 30days -> Continue ->

-> Run analysis on your project ->



```
*sonar-scanner - Notepad
File Edit View

#Configure here general information about the environment, such as SonarQube server
#No information about specific project should appear here

#----- Default SonarQube server
#sonar.host.url=http://localhost:9000

#----- Default source code encoding
#sonar.sourceEncoding=UTF-8

sonar.projectKey=EXP10_AEC
sonar.projectName=EXP10_AEC
sonar.projectVersion=1.0
#sonar.projectBaseDir=C:\SFIT
sonar.sources=C:\SONARSCANNER\sonar-scanner-5.0.1.3006-windows\conf
```

**11. Create Hello world Lambda function using Python.**

**12. Create Hello world Lambda function using Java.**

**13. Create Hello world Lambda function using Nodejs.**

-> Search Lambda -> Create Function -> Use a blueprint -> (Blueprint name) Hello world Python 3.7

(Execution role) -> Create a new role from AWS policy templates -> role name -> Create function

Test -> Configure test event -> Event name -> Edit event JSON

**14. Create AWS Lambda function to log "an object has been added" on adding the object to s3 bucket.**

```
import json
import boto3
s3=boto3.client('s3')
def lambda_handler(event,context):
    bucket="q14bucket"
    dataToUpload = {}
    dataToUpload['PID'] = '211121'
    dataToUpload['DEPT'] = 'INFT'
    dataToUpload['NAME'] = 'Brijraaj'
    dataToUpload['FILE'] = 'brij'
    fileName = 'brij' + '.json'
    uploadByteStream= bytes(json.dumps(dataToUpload).encode('UTF-8'))
    s3.put_object(Bucket=bucket,Key=fileName,Body=uploadByteStream)
    print('an object has been added')
```

Search IAM -> Roles -> Create role -> (Usecase) Lambda -> Next

**Permissions policies**

[CloudWatchFullAccess](#)

[AWSLambdaBasicExecutionRole](#)

[AmazonS3FullAccess](#)

->Enter Role Name -> Create Role

->Search S3 -> Create Bucket -> Enter Bucket name -> Create

->Search Lambda -> Create function -> Enter name -> Python 3.7 -> Change default execution role -> Use an existing role -> role1->Create function

->After creating paste the code

Click on Deploy -> click test -> Invoke

### **15.Create AWS Lambda function to visualise invocations.**

Search Lambda -> Create Function -> Use a blueprint -> (Blueprint name) Hello world Python 3.7

(Execution role) -> Create a new role from AWS policy templates -> role name -> Create function

Test -> Configure test event -> Event name -> Edit event JSON

Use invoke instead of run/save. Next go to monitor for visualization

### **16.Create AWS Lambda function to log "I got output".**

Search Lambda -> Create new function Python 3.7 -> test it

## **17. Create EC2 instance with following configurations- OS- Ubuntu (free tier)**

**instance type-t2.micro**

**key pair-.ppk**

**1. connect to the created instance**

**2. display present working directory**

-> Pwd

-> for superuser

sudo adduser <new-username>

sudo usermod -aG sudo <new-username>

sudo su - <new-username>

## **22. Deploy AWS Beanstalk environment**

Go to AWS console -> Search IAM user -> Create role -> (Usecase (EC2)) -> Next

In policies select the following

[AWSElasticBeanstalkWebTier](#)

-> Click on next -> Enter role name -> Create Role

Search ElasticBeanStalk -> Create Application -> In Configure Environment -> Enter name of application -> Scroll down to Choose platform -> Select PHP -> Next

In service -> Select Existing Role -> Select Role 4 -> In EC2 instance profile select Role4->Next



## Set up networking, database, and tags - *optional* [Info](#)

### Virtual Private Cloud (VPC)

#### VPC

Launch your environment in a custom VPC instead of the default VPC. You can create a VPC and subnets in the VPC management console.

[Learn more](#) 

vpc-0aa5774356fe7db79 | (172.31.0.0/16) ▼

[Create custom VPC](#) 

Select VPC -> Scroll down -> Select US-east-1-a (Instance Subnet) -> Next

Scroll down -> EC security groups -> Select Default -> Next

Next -> Submit

Open AWS in new tab -> Search Code Pipeline -> Create Pipeline -> Name pipeline -> V1

-> New Service Role -> Next

Add source stage

-Github (version 2) -> Connect to github -> -> Give connection some name -> connect to GitHub  
-> Authorize AWS connector for GitHub -> -> Install a new app -> Only select Repositories ->  
Select the repository -> Install -> Connect -Ready to connect Notification -> Select GitHub  
repository you want to use as the source location -> Branch name: master -> CodePipeline  
default -> Next

-> Skip build stage

Skip

-> Deploy provider -> Elastic Beanstalk -> Default Region -> appname -> env name -> next

Activate pipeline -> Review and create pipeline