

1. VPC peer from your team's VPC to the next team i.e. A->B, B->C, ..., E->A.
 - Allow both directions VPC Traffic.
 - Add route to peered VPC in private Route Table.

In intern C region, “peer connection” tab was selected and new peer connection was created:

Select a local VPC to peer with

VPC ID (Requester)

vpc-058da4e261ea01687 (intern-c-vpc) ▼

VPC CIDRs for vpc-058da4e261ea01687 (intern-c-vpc)

CIDR	Status	Status reason
10.15.24.0/22	✔ Associated	-

Select another VPC to peer with

Account

☒ My account

☐ Another account

Region

☐ This Region (us-east-1)

☒ Another Region

US East (Ohio) (us-east-2) ▼

VPC ID (Acceptor)

vpc-0537baf72f80d5930

Peering connections (1/1) Info						
Filter peering connections						
Name	Peering connection ID	Status	Requester VPC	Acceptor VPC	Requester	
–	pcx-0097be4c01f4acce	✔ Active	vpc-058da4e261ea01687	vpc-0537baf72f80d5930 / Te...	10.15.24.0	

In intern-d region: goto to peer connection > action > Accept request

DNS | Route tables | Tags

DNS settings [Edit DNS settings](#)

<p>Requester VPC (vpc-058da4e261ea01687 / intern-c-vpc) Info</p> <p>Allow acceptor VPC to resolve DNS of hosts in requester VPC to private IP addresses</p> <p>⊖ Disabled</p>	<p>Acceptor VPC (vpc-0537baf72f80d5930) Info</p> <p>Allow requester VPC to resolve DNS of hosts in acceptor VPC to private IP addresses</p> <p>⊖ Disabled</p>
--	--

🟢 A VPC peering connection pcx-0097be4c01f4facce / peering-vpc-CD has been requested. Remember to change your region to us-east-2 to accept the peering connection.

VPC > Peering connections > pcx-0097be4c01f4facce

pcx-0097be4c01f4facce / peering-vpc-CD

Actions ▾

Details Info

Requester owner ID 🔑 949263681218	Accepter owner ID 🔑 949263681218	Peering connection ID 🔑 pcx-0097be4c01f4facce
Requester VPC vpc-058da4e261ea01687 / intern-c-vpc	Accepter VPC vpc-0537baf72f80d5930	Status 🕒 Initiating Request to 949263681218
Requester CIDRs 📏 10.15.24.0/22	Accepter CIDRs -	Expiration time Tuesday, December 14, 2021, 00:53:05 GMT+5:45
Requester Region N. Virginia (us-east-1)	Accepter Region Ohio (us-east-2)	

🟢 Your VPC peering connection (pcx-0097be4c01f4facce) has been established. To send and receive traffic across this VPC peering connection, you must add a route to the peered VPC in one or more of your VPC route tables. Info

Modify my route tables now ✕

VPC > Peering connections > pcx-0097be4c01f4facce

pcx-0097be4c01f4facce

Actions ▾

Details Info

Requester owner ID 🔑 949263681218	Accepter owner ID 🔑 949263681218	Peering connection ID 🔑 pcx-0097be4c01f4facce
Requester VPC 🔑 vpc-058da4e261ea01687	Accepter VPC vpc-0537baf72f80d5930 / Team-D-VPC	Status 🕒 Provisioning
Requester CIDRs 📏 10.15.24.0/22	Accepter CIDRs -	Expiration time -
Requester Region N. Virginia (us-east-1)	Accepter Region Ohio (us-east-2)	

Add route to peered VPC in private Route Table.

Intern c > route table > route table private > route>edit
put the ip of intern-d privateroutetable and peerconnection-id

VPC > Route tables > rtb-09f9cee465488e9a1 > Edit routes

Edit routes

Destination	Target	Status	Propagated	
pl-7ba54012	vpce-053438f7b12054ea6	🟢 Active	No	
10.15.32.0/22	🔍 local ✕	🟢 Active	No	
🔍 0.0.0.0/0 ✕	🔍 nat-0d9875d702c83b8f9 ✕	🟢 Active	No	Remove
🔍 10.15.24.0/22 ✕	🔍 pcx- pcx-0097be4c01f4facce ✕	-	No	Remove

Add route

pcx-0097be4c01f4facce

Cancel Preview Save changes

Updated routes for rtb-09f9cee465488e9a1 / Team-D-Pvt-RTB successfully

Details

VPC > Route tables > rtb-09f9cee465488e9a1

rtb-09f9cee465488e9a1 / Team-D-Pvt-RTB

Actions

You can now check network connectivity with Reachability Analyzer

Run Reachability Analyzer

Details Info

Route table ID

rtb-09f9cee465488e9a1

VPC

vpc-0537baf72f80d5930 | Team-D-VPC

Main

No

Owner ID

949263681218

Explicit subnet associations

3 subnets

Edge associations

-

Routes

Subnet associations

Edge associations

Route propagation

Tags

Do same as vice versa from internD

Updated routes for rtb-03da9f2d37e6ca299 / intern-c-routetable-pvt successfully

Details

rtb-03da9f2d37e6ca299

No

3 subnets

-

VPC

vpc-058da4e261ea01687 | intern-c-vpc

Owner ID

949263681218

Routes

Subnet associations

Edge associations

Route propagation

Tags

Routes (4)

Edit routes

Filter routes

Both

< 1 >

Destination	Target	Status	Propagated
10.15.24.0/22	local	Active	No
10.15.32.0/22	pcx-0097be4c01f4facce	Active	No
0.0.0.0/0	nat-0177c13ef7175fb50	Active	No
pl-63a5400a	vpce-0afca0e8e20c44634	Active	No

2. Create RDS cluster and instance of PostgreSQL.

- Deploy in a private subnet.
- In your private EC2. Connect to the RDS Postgresql.
- (Optional) Validate you can connect to another team's (A->B and B->C) RDS (Peer must be completed)

RDS cluster was created by selecting Services>Database>RDS

And selected DB instances and created new instances of Postgresql.

Create database

Choose a database creation method [info](#)

- ☐ Standard create
You set all of the configuration options, including ones for availability, security, backups, and maintenance.
- ☒ Easy create
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Configuration

Engine type [info](#)

- ☐ Amazon Aurora
- ☐ MySQL
- ☐ MariaDB
- ☒ PostgreSQL
- ☐ Oracle
- ☐ Microsoft SQL Server

Instance type

- ☐ Production
db.r6g.xlarge
4 vCPUs
32 GiB RAM
500 GiB
1.057 USD/hour
- ☒ Dev/Test
db.r6g.large
2 vCPUs
16 GiB RAM
100 GiB
0.241 USD/hour

DB instance identifier
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.
intern-c-psql-instance

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

Master username [info](#)
Type a login ID for the master user of your DB instance.
postgres

1 to 16 alphanumeric characters. First character must be a letter.

☒ **Auto generate a password**
Amazon RDS can generate a password for you, or you can specify your own password.

Master password [info](#)
Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), ' (single quote), " (double quote) and @ (at sign).

Confirm password [info](#)

After providing an instance identifier name, username and password for our database, the database instance is finally created after a few minutes.

Creating database intern-c-psql-instance
Your database might take a few minutes to launch. [View credential details](#)

RDS > Databases

Databases [Group resources](#) [Refresh](#) [Modify](#) [Actions](#) [Restore from S3](#) [Create database](#)

	DB identifier	Role	Engine	Region & AZ	Size	Status	CPU	Current activity	Maintenan
<input type="radio"/>	intern-c-psql-instance	Instance	PostgreSQL	-	db.r6g.large	⌚ Creating	-		none
<input type="radio"/>	intern-tutor	Instance	PostgreSQL	us-east-1d	db.t3.micro	✅ Available	-		none

3. Install AWS CLIV2 and configure the lft-training profile.

AWS CLI was installed in our host machine simply by using some commands:

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

```
saroj@saroj-Inspiron-3576:~/Downloads/awscli$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total       Spent    Left     Speed
18 43.3M   18 8367k   0      0  272k    0  0:02:42  0:00:30  0:02:12 292k
```

unzip awscliv2.zip

sudo ./aws/install

```
inflating: aws/dist/cryptography-3.3.2-py3.8.egg-info/LICENSE
inflating: aws/dist/cryptography-3.3.2-py3.8.egg-info/WHEEL
creating: aws/dist/cryptography/hazmat/
creating: aws/dist/cryptography/hazmat/bindings/
inflating: aws/dist/cryptography/hazmat/bindings/_openssl.abi3.so
bj@batman:~/aws$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
```

After aws is installed in our system, we can verify using command:

Aws --version

To configure the profile, we can start using these commands:

Aws configure

After entering our aws access key id and secret access key with region name and output format as json, we configured our profile for aws.

```
bj@batman:~/aws$ aws configure
AWS Access Key ID [None]: AKIA52BEGI3BCCP6PEYC
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
```

```
saroj@saroj-Inspiron-3576:~/Downloads/awscli$ aws configure
AWS Access Key ID [None]: AKIA52BEGI3BCCP6PEYC
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
saroj@saroj-Inspiron-3576:~/Downloads/awscli$
```

```
saroj@saroj-Inspiron-3576:~/Downloads/awscli$ aws ecr get-login-password --region us-east-1 | sudo docker login --username AWS --password-stdin 949263681218.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
saroj@saroj-Inspiron-3576:~/Downloads/awscli$
```

4. Create ECR and upload your Docker image created during Docker assignment Q3.
 - Each member must upload an image (in the team's ECR repo) with their name as tag.

Now, to login to our aws system: `aws ecr get-login-password --region us-east-1 | sudo docker login --username AWS --password-stdin 949263681218.dkr.ecr.us-east-1.amazonaws.com`

```
bj@batman:~/aws$ aws ecr get-login-password --region us-east-1 | sudo docker login --username AWS --password-stdin 949263681218.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

And we can see the login has succeeded.

Now we pull docker images from our individual docker hub accounts which we have uploaded in a previous assignment of docker. All our team members did the same with our individual docker images from docker hub.







`sudo docker pull bijaykandel37/nodedocker:nodedocker`

`sudo docker tag bijaykandel37/nodedocker:nodedocker
949263681218.dkr.ecr.us-east-1.amazonaws.com/intern-c:bijaykandel37`

`sudo docker push 949263681218.dkr.ecr.us-east-1.amazonaws.com/intern-c:bijaykandel37`

```
bj@batman:~/aws$ sudo docker pull bijaykandel37/nodedocker:nodedocker
nodedocker: Pulling from bijaykandel37/nodedocker
97518928ae5f: Pull complete
7001f79e6409: Pull complete
ad6524892285: Pull complete

bj@batman:~/aws$ sudo docker tag bijaykandel37/nodedocker:nodedocker 949263681218.dkr.ecr.us-east-1.amazonaws.com/intern-c:bijaykandel37
bj@batman:~/aws$ docker push 949263681218.dkr.ecr.us-east-1.amazonaws.com/intern-c:bijaykandel37
```

<input type="checkbox"/>	Image tag	Pushed at	Size (MB)	Image URI	Digest	Scan status	Vulnerabilities
<input type="checkbox"/>	bijaykandel37	December 07, 2021, 18:50:58 (UTC+05.75)	81.72	 Copy URI	 sha256:5407566781915c...	-	-
<input type="checkbox"/>	saroj	December 07, 2021, 11:57:09 (UTC+05.75)	56.73	 Copy URI	 sha256:b41876afcb9bfae...	-	-
<input type="checkbox"/>	prajesh	December 07, 2021, 11:55:56 (UTC+05.75)	50.69	 Copy URI	 sha256:6241e88cc42c62...	-	-

5. Deploy ***one*** of uploaded image in ECS Fargate.

- Create task definition.
- Deploy in Public Subnet.
- Access via assigned Public IP and your port.

New task definition was created by selecting ECS and then Task Definitions:


Create new Task Definition

Step 1: Select launch type compatibility
Step 2: Configure task and container definitions

Select launch type compatibility


Select which launch type you want your task definition to be compatible with based on where you want to launch your task.

FARGATE




Price based on task size
Requires network mode awsvpc
AWS-managed infrastructure, no Amazon EC2 instances to manage

EC2



Price based on resource usage
Multiple network modes available
Self-managed infrastructure using Amazon EC2 instances

EXTERNAL



Price based on instance-hours and additional charges for

And we added our container with a proper link to our deployed container.

Add container

Standard

Container name* intern-c-container ⓘ

Image* 949263681218.dkr.ecr.us-east-1.amazonaws.com/intern-c:bijaykandel37 ⓘ

Private repository authentication* ☐ ⓘ

Memory Limits (MiB) Soft limit 128 ⓘ
[+ Add Hard limit](#)

Launch Status

Task definition status - 3 of 3 completed

Create Execution Role

Execution Role AmazonECSTaskExecutionRole created [Learn more](#)

Create Task Definition: intern-c-fargate-task

intern-c-fargate-task succeeded

Create CloudWatch Log Group

✔ **CloudWatch Log Group** created
CloudWatch Log Group [/ecs/intern-c-fargate-task](#)

Now the task definition is created which can be seen by going to Tasks tab:

<input type="checkbox"/>	Task Definition	Latest revision status
<input type="checkbox"/>	intern-a	ACTIVE
<input type="checkbox"/>	intern-c-fargate-task	ACTIVE
<input type="checkbox"/>	intern-tutor	ACTIVE

To run the task, we selected the task and Actions>RunTask.

Run Task

Select the cluster to run your task definition on and the number of copies of that task to run. To apply container o

Launch type

☒ FARGATE
☐ EC2
☐ EXTERNAL

Switch to capacity provider strategy

Operating system family

Task Definition

Family

intern-c-fargate-task

Enter a value

Revision

1 (latest)

Platform version

LATEST

Cluster

intern-c

Number of tasks

1

VPC and security groups

VPC and security groups are configurable when your task definition uses the awsvpc network mode.

Cluster VPC*

vpc-058da4e261ea01687 (10.15.24.0/22...)

Subnets*

subnet-0adc7dac5cc424e4e (10.15.24.0/27) | intern-c-pub-1 - us-east-1

a

assign ipv6 on creation: Disabled

Security groups*

intern-1885

Edit

Auto-assign public IP

ENABLED

Advanced Options

Task tagging configuration

☒ Enable ECS managed tags

Propagate tags from

Do not propagate

Tags

After providing all necessary details like Task definition name, cluster and cluster VPC. Also a Security Group was created for this specific app and we opened our exposed port which was 8787 in our case of the node container.

Now the task is running and we can get the public IP by going to Tasks tab and Under Network section, we can find public ip:

Cluster

intern-c

Launch type

FARGATE

Platform version

1.4.0

Task definition

intern-c-fargate-task:1

Group

family:intern-c-fargate-task

Task role

None

Last status

RUNNING

Desired status

RUNNING

Created at

2021-12-08 12:10:12 +0545

Started at

2021-12-08 12:10:52 +0545

Network

Network mode

awsvpc

ENI Id

eni-0bb3fe699d6f557d4

Subnet Id

subnet-0adc7dac5cc424e4e

Private IP

10.15.24.21

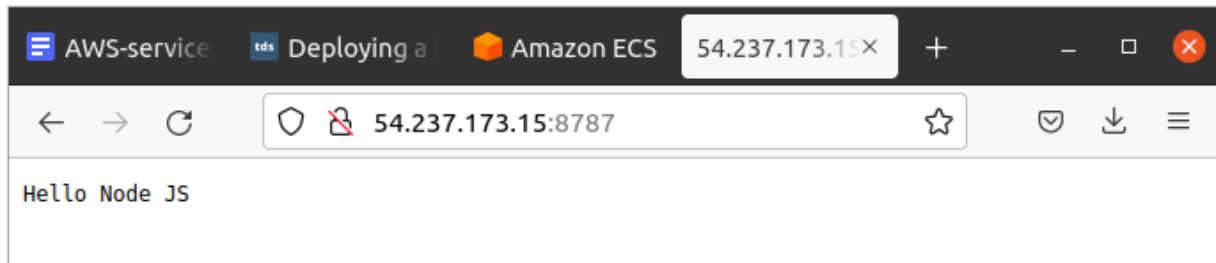
Public IP

54.237.173.15

Mac address

0e:f4:f3:92:f1:2f

And in our browser, when we use the IP with our port address, the Node instance is LIVE and fully accessible on the internet.




6. Create a S3 bucket

- Upload Dockerfile of Q4 in the bucket using AWS CLI.

Bucket was created by selecting the S3 tab and then 'Create bucket' option on top right with appropriate information and Selecting Create bucket option at bottom.

We can see the bucket we created under the buckets section in S3.

 intern-c-bucket	US East (N. Virginia) us-east-1	Bucket and objects not public	December 6, 2021, 22:17:12 (UTC+05:45)
---	------------------------------------	----------------------------------	---

Now, Using aws cli, we can list all buckets with command:

Aws s3 ls

And upload any docker file to a bucket using command:

Aws s3 cp /path/to/Dockerfile s3://name-of-bucket

In our case:

Aws s3 cp Dockerfile s3://intern-c-bucket







```
bj@batman:~/Documents/AssignmentsLftech/4_4_server-docker-krishna-bijaykandel37/3/nginx$ aws s3 ls
2021-12-08 11:30:36 aws-cloudtrail-logs-949263681218-62532415
2021-12-06 23:24:29 intern-a
2021-12-06 22:17:12 intern-c-bucket
2021-12-06 11:54:55 intern-tutor
2021-12-08 15:04:23 team-5-s3-dockerfile-bucket
2021-12-05 15:48:56 team-b-s3-bucket
2021-12-07 17:34:22 team-d-s3-bucket
bj@batman:~/Documents/AssignmentsLftech/4_4_server-docker-krishna-bijaykandel37/3/nginx$ ls
build.png Dockerfile hosted.png index.html nginxdockerfile.png
bj@batman:~/Documents/AssignmentsLftech/4_4_server-docker-krishna-bijaykandel37/3/nginx$ aws s3 cp Dockerfile s3://intern-c-bucket/
upload: ./Dockerfile to s3://intern-c-bucket/Dockerfile
```

Now, we can see the uploaded Dockerfile in S3 section and inside our specified bucket:

intern-c-bucket [Info](#)


[Objects](#) | [Properties](#) | [Permissions](#) | [Metrics](#) | [Management](#) | [Access Points](#)


Objects (1)
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)


  Copy S3 URI  Copy URL  Download  Open  Delete

Actions ▾

Create folder

 Upload

< 1 > 

<input type="checkbox"/>	Name ▲	Type ▼	Last modified ▼	Size ▼	Storage class ▼
<input type="checkbox"/>	 Dockerfile	-	December 8, 2021, 18:42:48 (UTC+05:45)	54.0 B	Standard