

## מדריך משתמש AWS Cloud9 – שפת C

### 1. היכרות עם סביבת העבודה AWS Cloud9

האתר: <https://aws.amazon.com/cloud9/>



יש להירשם לאתר (Create an AWS Account)

### Create an AWS account

Email address

Password

Confirm password

AWS account name ⓘ

Continue

ולאחר מכאן אפשר להיכנס (Sign in) ל-AWS account



## Sign in ⓘ

Email address of your AWS account

Or to sign in as an IAM user, enter your account ID or [account alias](#) instead.

Next

Cloud9 ← Developer Tools

יוצרים environment חדש ע"י Create environment. נותנים שם ל-environment חדש, מבצעים הגדרות בהתאם:

### Environment settings

#### Environment type [Info](#)

Choose between creating a new EC2 instance for your new environment or

- ☒ Create a new instance for environment (EC2)  
Launch a new instance in this region to run your new environment.
- ☐ Connect and run in remote server (SSH)  
Display instructions to connect remotely over SSH and run your new environment.

#### Instance type

- ☒ t2.micro (1 GiB RAM + 1 vCPU)  
Free-tier eligible. Ideal for educational users and exploration.
- ☐ t2.small (2 GiB RAM + 1 vCPU)  
Recommended for small-sized web projects.
- ☐ m4.large (8 GiB RAM + 2 vCPU)  
Recommended for production and general-purpose development.
- ☐ Other instance type  
Select an instance type.

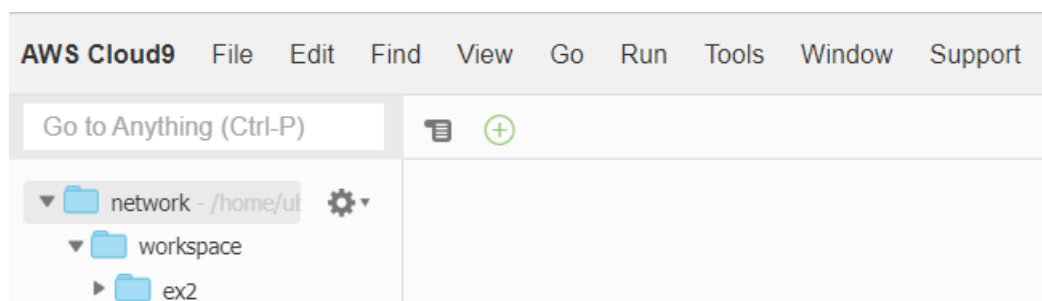
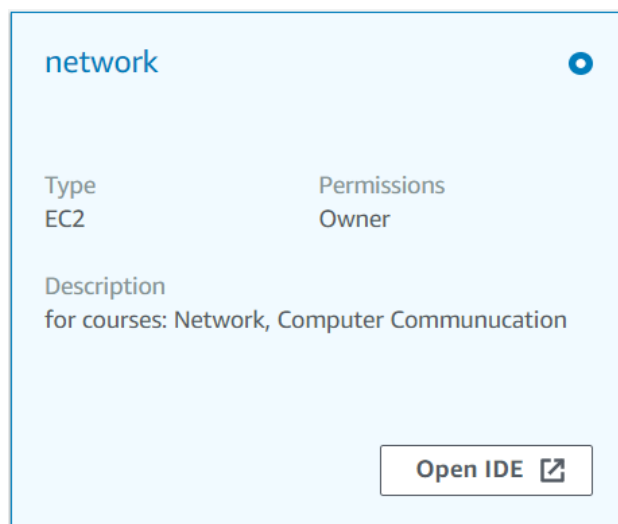
t3.nano

#### Platform

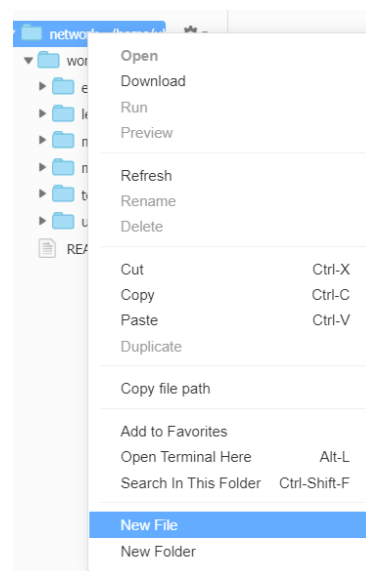
- ☐ Amazon Linux
- ☒ Ubuntu Server 18.04 LTS

לאחר מכאן, ע"י הקלקה על Open IDE מגיעים ל-environment שיצרנו:

## מדריך משתמש AWS Cloud9 – שפת C

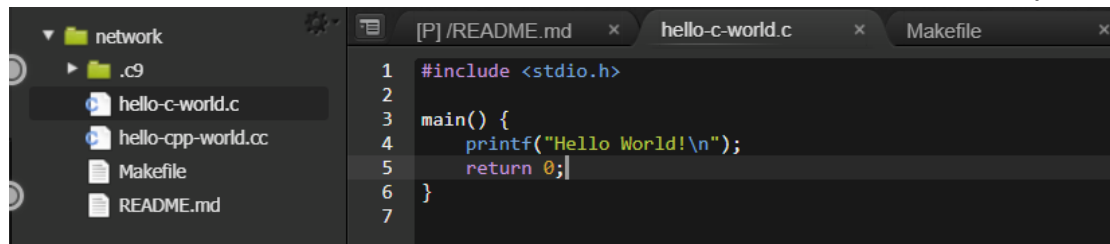


ניתן ליצור תיקיות, קבצים תחת ה-environment שיצרנו ע"י קליק ימני.



## מדריך משתמש AWS Cloud9 – שפת C

יוצרים קובץ חדש ונותנים לו שם לדוגמה: `name.c` (לא לשכוח סיומת `c`). ובקובץ זה כותבים את הקוד.



### קומפילציה:

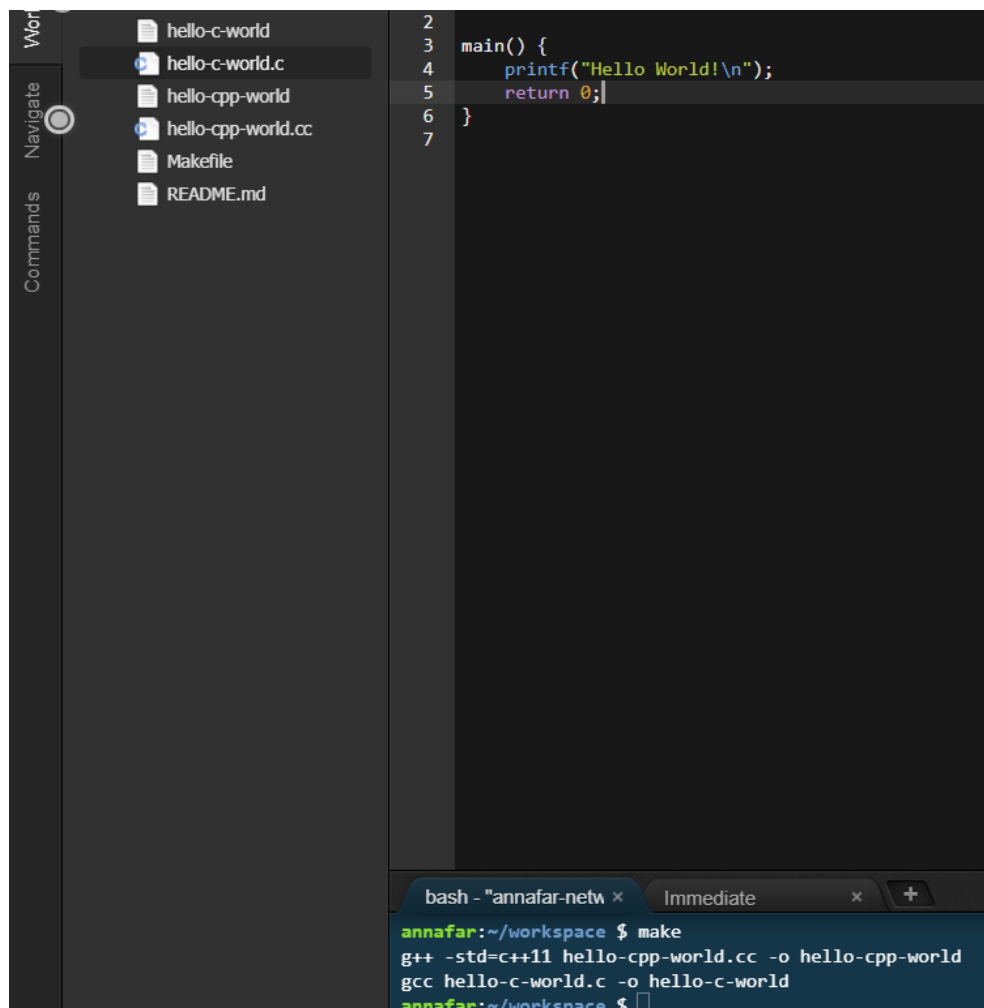
כאשר רוצים לקמפל, רושמים ב Terminal:

```
gcc -o file_name name.c
```

או

```
gcc name.c -o file_name
```

לאחר הקימפול, נוצר קובץ חדש עם השם שרשמנו.



כתבה: אנה פרבר

## מדריך משתמש AWS Cloud9 – שפת C

כדי להריץ את הקובץ שנוצר נרשום ב Terminal:

`./ file_name`

```
gcc hello-c-world.c -o hello-c-world
annafar:~/workspace $ ./hello-c-world
Hello World!
annafar:~/workspace $
```

אם רוצים לראות את ה-warnings שבקוד יש להוסיף לשורת הקימפול: -Wall

`gcc -Wall name.c -o file_name`

וכאשר רוצים להפוך את ה-warnings ל-error יש להוסיף: -Werror

`gcc -Wall -Werror name.c -o file_name`

דוגמא: ללא -Wall

```
gcc hello-c-world.c -o hello-c-world
annafar:~/workspace $ ./hello-c-world
```

דוגמא: עם -Wall

```
annafar:~/workspace $ gcc -Wall -Werror hello-c-world.c -o hello-c-world
hello-c-world.c:3:1: error: return type defaults to 'int' [-Werror=return-type]
main() {
^
cc1: all warnings being treated as errors
annafar:~/workspace $
```

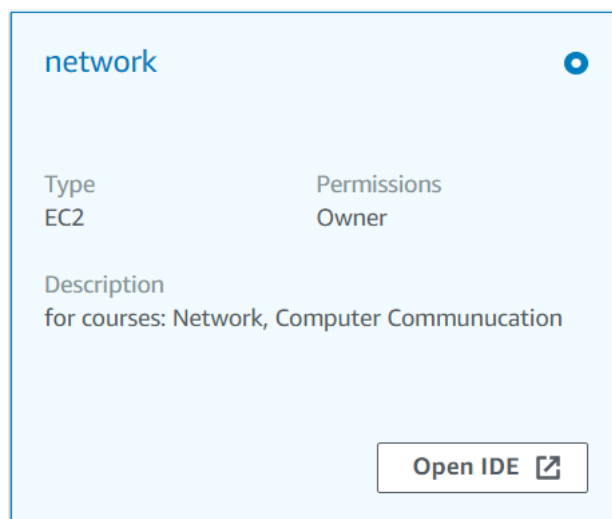
למידע נוסף ופקודות נוספות:

<http://www.rapidtables.com/code/linux/gcc.htm>

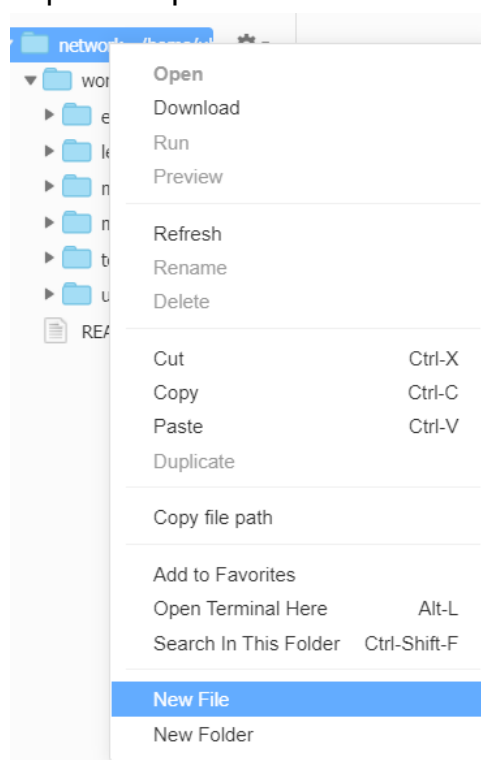
## 2. הרצת קוד נתון (מודל server-client)

ע"י הקלקה על Open IDE מגיעים ל-environment שיצרנו:

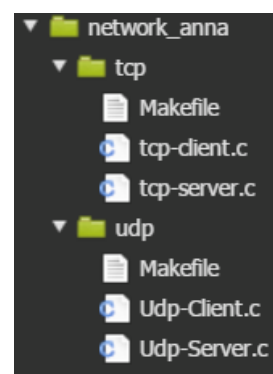
## מדריך משתמש AWS Cloud9 – שפת C



תחת ה-environment ניתן ליצור תיקיה חדשה (לדוגמא, matala\_4):



פותחים קבצי ה-zip הנתונים, מעתיקים את התיקיות (tcp, udp) תחת ה-environment:



כתבה: אנה פרבר

## מדריך משתמש AWS Cloud9 – שפת C

**Makefile** הוא קובץ שמכיל סט הוראות/פקודות לפקודה `make` ב-`command line`. שילוב פקודת `make` עם קובץ `Makefile` מייעל ומקצר תהליך הידור, מחיקת קבצי `out` ופעולות אחרות. בהפעלת `make` (כתיבת פקודה ב-`command line`), פעולה ראשונה שנעשה הינה חיפוש `Makefile` בתיקייה הנוכחית. על מנת להפעיל פקודה מקובץ `Makefile` יש לכתוב `שם פקודה` `make`.

להלן דוגמא לקובץ ה-**Makefile** של תיקיית `udp`:

```
# Makefile for UDP project
```

הערות

שם הפקודה/target

```
all: udp-server udp-client
```

תלות/dependencies

```
udp-server: Udp-Server.c
```

```
gcc -o Udp-Server Udp-Server.c
```

תוכן הפקודה/rules

tab

```
udp-client: Udp-Client.c
```

```
gcc -o Udp-Client Udp-Client.c
```

```
clean:
```

```
rm -f *.o Udp-Server Udp-Client
```

```
runs:
```

```
./Udp-Server
```

```
runc:
```

```
./Udp-Client
```

```
runs-strace:
```

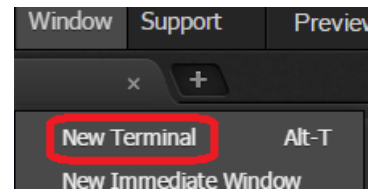
```
strace -f ./Udp-Server
```

```
runc-strace:
```

```
strace -f ./Udp-Ccdlient
```

פותחים שני `terminals`: אחד עבור הרצת שרת ושני עבור הרצת לקוח:

## מדריך משתמש AWS Cloud9 – שפת C



להלן דוגמא להרצת המערכת בשימוש קובץ Makefile:

\*לא לשכוח לעבור לתיקייה המתאימה (במקרה שלנו, cd udp).

לפני הרצה יש לוודא מה IP השרת ע"י פקודה ifconfig ולעדכן את הקבצים בהתאם:

```
#define SERVER_IP_ADDRESS "127.0.0.1"
```

פקודות בטרמינל של שרת:

```
bash - "annafar-netw" x  make - "ubuntu@ann" x +
annafar:~/workspace/tcp $ cd ../
annafar:~/workspace $ cd udp
annafar:~/workspace/udp $ make
gcc -o Udp-Server Udp-Server.c
gcc -o Udp-Client Udp-Client.c
Udp-Client.c: In function 'main':
Udp-Client.c:113:2: warning: format not a string literal and no format arguments [-Wformat-security]
    printf(bufferReply);
    ^
annafar:~/workspace/udp $ make runs
./Udp-Server
After bind(). Waiting for clients
```

פקודות בטרמינל של לקוח:

```
bash - "annafar-netw" x  make - "ubuntu@ann" x +
annafar:~/workspace $ cd udp
annafar:~/workspace/udp $ make runc
./Udp-Client
Hello, from the Server
annafar:~/workspace/udp $
```

בסיום הרצה לא לשכוח למחוק object files ו-executable files ע"י הפקודה הבאה (לא משנה באיזה טרמינל להריץ את הפקודה):

```
annafar:~/workspace/udp $ make runs
./Udp-Server
After bind(). Waiting for clientsReceived packet from 127.0.0.1:48252
Data is: Good morning, Vietnam

^Cmake: *** [runs] Interrupt

annafar:~/workspace/udp $ make clean
rm -f *.o Udp-Server Udp-Client
annafar:~/workspace/udp $
```

על ידי פקודה strace ניתן לעשות debugging לקוד שרץ:



## מדריך משתמש AWS Cloud9 – שפת C

בטרמינל של שרת:

```
annafar:~/workspace/udp $ make runs-strace  
strace -f ./Udp-Server
```

בטרמינל של לקוח:

```
annafar:~/workspace/udp $ make runc-strace  
strace -f ./Udp-Ccdlient
```