¿Cómo configuro mi instancia de EC2
Para configurar EC2 lo que debes de hacer es:

1. Crear una cuenta en Amazon Web Services. Si es tu primera vez usando AWS tendrás acceso a la versión gratuita que se podrá usar el primer año https://aws.amazon.com/es/
2. Dirigirte al servicio EC2
3. Ahora las configuraciones básicas que se deben de hacer son:
    a. Indicar el sistema operativo que se utilizará. Recomiendo usar Ubuntu ya que este es compatible con NGINX que será necesario para configurar el balanceo de carga y HTTPS. Además es bastante barato en cuanto a su costo.
    b. Luego es necesario indicar las especificaciones que tendrá la instancia. Si solo se cuenta con versión gratuita se recomienda usar t2.micro o t3.micro. Por otro lado, es necesario tener en cuenta los frameworks que utilizarán en el equipo ya que algunos framework como Next.js pueden necesitar de mayor memoria para compilar el proyecto. Si se usa más memoria puede resultar más cómodo utilizar instancias de pago con mayor memoria.
    c. Después es necesario indicar si se crearán nuevas claves (.pem o .ppk) o si se utilizarán claves ya existentes. Es importante guardar estas claves ya que solo se podrá conectar a la instancia mediante estas.
    d. Posteriormente, tendrá que configurar los puertos aqui es recomendable dejar el puerto SSH, por defecto activado, como está. Luego marcar los checkbox para abrir tanto el puerto http y el puerto https. Es recomendable solo abrir estos puertos si el sistema desarrollado ya cuenta con autenticación.
    e. Finalmente indicar la cantidad de espacio en disco que se requerirá. Se puede máximo 30GB si es gratuito y se cuentan todas las instancias que posee la cuenta de la persona.
4. Una vez terminado solo se tendra que hacer click en lanzar instancia

¿Qué recomendaciones se deben de tener al usar instancias EC2?
Lo que se tiene que tener en cuenta para usar instancias EC2 es que uno se puede apoyar de herramientas como github para extraer lo necesario para que el sistema pueda desplegarse. Entonces si se trabaja usando python y Typescript para back y front respectivamente
Configurar .git ignore para no subir lo siguiente:

```
# ==============================
# FRONTEND CONFIG (Node.js / React / Next.js)
# ==============================

# Node.js dependencies
node_modules/
package-lock.json
pnpm-lock.yaml
yarn.lock
.next/

# TypeScript build artifacts
```

```
dist/
build/
out/
*.tsbuildinfo

# Logs
logs/
*.log
npm-debug.log*
yarn-debug.log*
pnpm-debug.log*
lerna-debug.log*

# Environment variables
.env
.env.local
.env.development
.env.test
.env.production
.env.development.local
.env.test.local
.env.production.local
datacargable/

# Editor & OS files
.vscode/
!.vscode/extensions.json
.idea/
.DS_Store
*.suo
*.ntvs*
*.njsproj
*.sln
*.sw?

# Jest testing
coverage/
jest-cache/
jest-stare/

# Linting & formatting
eslintcache
.prettier-cache/
stylelintcache
```

```
# ================================
# BACKEND CONFIG (Python / Django)
# ================================

# Python cache & compiled files
__pycache__/
*.py[cod]
*$py.class

# Virtual environments
.env
.venv/
env/
venv/
ENV/
env.bak/
venv.bak/

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
.eggs/
*.egg-info/
.installed.cfg
*.egg
MANIFEST

# Installer logs
pip-log.txt
pip-delete-this-directory.txt

# Unit test / coverage reports
htmlcov/
.tox/
.nox/
.coverage
.coverage.*
.cache
nosetests.xml
coverage.xml
```

```
*.cover
*.py,cover
.hypothesis/
.pytest_cache/
cover/

# Django specific
*.log
*.pyc
local_settings.py
db.sqlite3
db.sqlite3-journal
staticfiles/  # Archivos estáticos recolectados en producción
media/        # Archivos subidos por usuarios


# Jupyter Notebook
.ipynb_checkpoints

# PyCharm & VS Code settings
.vscode/
.idea/
.spyderproject
.spyproject
.ropeproject

# Type checking tools
.mypy_cache/
.pyre/
.pytype/
.dmypy.json
dmypy.json

#test
*.rest
```

En la instancia sólo será necesario ejecutar la descarga de todas las dependencias para el proyecto. Esto puede realizarse sin problemas con herramientas como npm install y pip install -r requirements.txt. Después realizar la compilación del proyecto. De esta manera, no será necesario subir volúmenes grandes de datos a github.

¿Cómo uso rds para mis proyectos de dp1 y lp2?

Con respecto a RDS para los trabajos de dp1 y lp2 no será necesario tener redundancia por lo que bastará con establecer solo una base de datos. Para ello se usará el sistema RDS de amazon web services (AWS).

1. Buscar RDS
2. Hacer click en create database
3. Tener la opción dde standard create
4. Escoger el administrador de base de datos con el que se trabajará en el proyecto. Por ejemplo para python con django se recomienda usar PostgreSQL ya que MySql no es completamente compatible con sistemas linux. Luego escoger la versión deseada.
5. Escoger la plantilla necesaria para el proyecto. Si se tiene una cuenta gratuita free tier por ejemplo.
6. Para las opciones de disponibilidad, como los proyectos no necesitan de alta disponibilidad ya que no serán puestos en producción se recomienda Single DB instance
7. En DB instace identifier indicar el nombre de la base de datos
8. En Master username especificar el usuario maestro
9. Indicar master password para acceder como administrador a la base de datos
10. Después dirigirse a la sección storage donde se indicará el espacio que se desea tener

¿Por qué se recomienda usar archivos .env?
Porque es importante ocultar información sensible como las claves secretas del proyecto ya sea backend y frontend. También es importante ocultar contraseñas y direcciones de la base de datos. Entonces los archivos .env son especialmente útiles para estos casos ya que son interpretados como archivos ocultos por el sistema lo que evita su acceso sin cuidados. Es importante no incluir estos archivos en el repositorio ya que serán registrados en el historial de GIT, esto puede ocasionar que si en caso el proyecto se hace público el resto de personas puede acceder a información sensible.

¿Para qué sirven los virtual environment?
Los virtual environment son herramientas muy útiles para un proyecto porque permiten no tener que instalar todos los paquetes directamente en el computador de desarrollo sino en un entorno limpio. De esta manera, se evitan conflictos entre librerías. Además, se puede guardar el nombre de las librerías usadas y luego instalarlas en otro ordenador como puede ser la instancia de amazon . Existen diferentes formas de instalarlas. Para lenguajes como python se puede usar virtualenv que permite crear este entorno virtual junto con el comando freeze que muestra el nombre de todas las dependencias existentes en el espacio virtual.