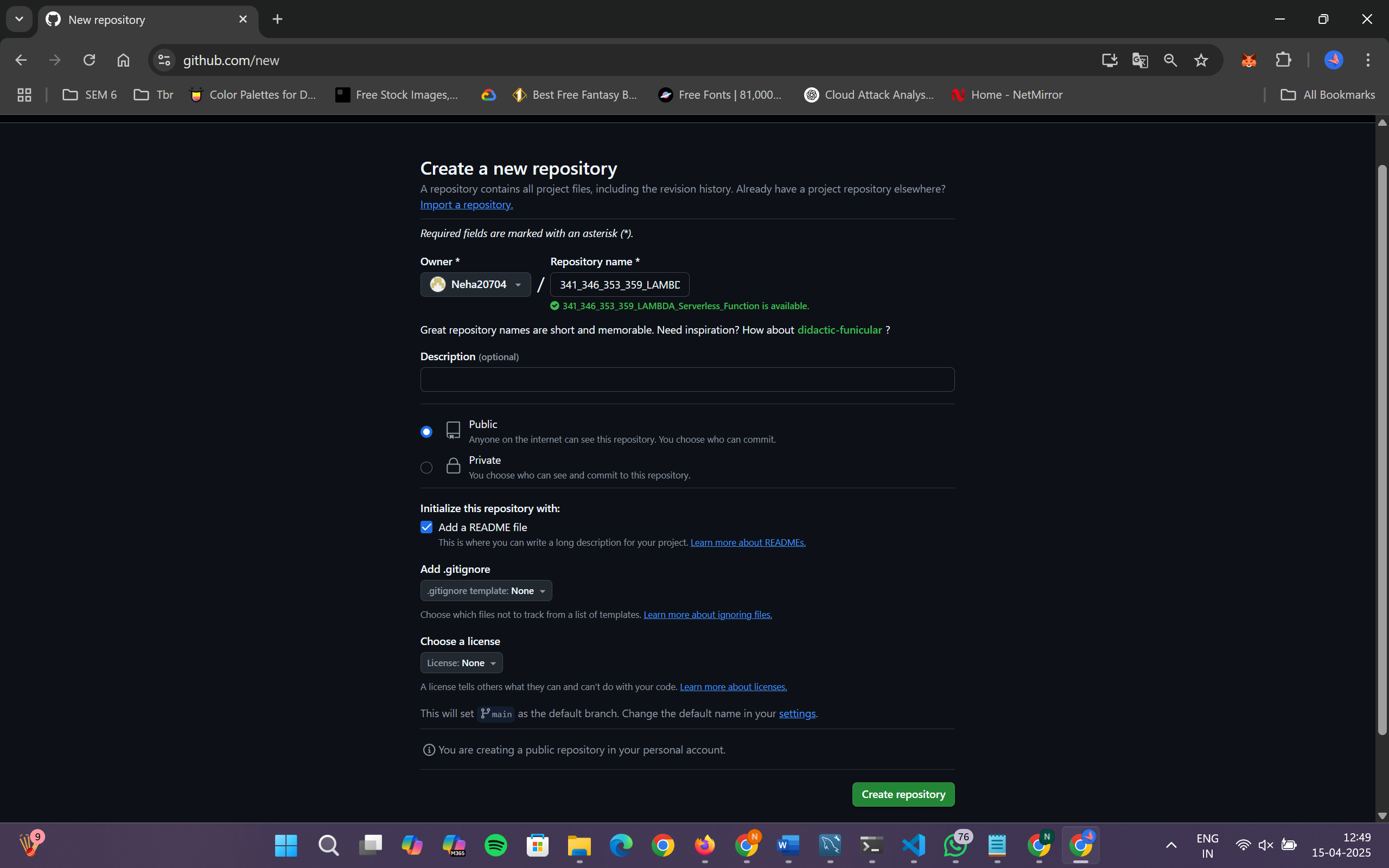
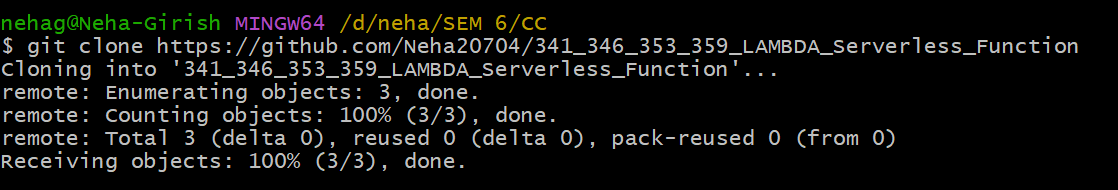
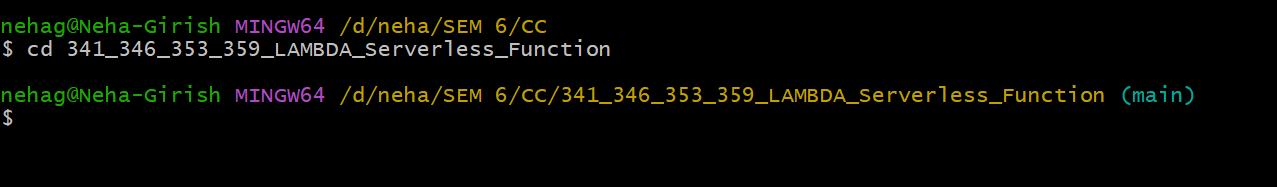
Creating a git repo and initializing it with a ReadME.md

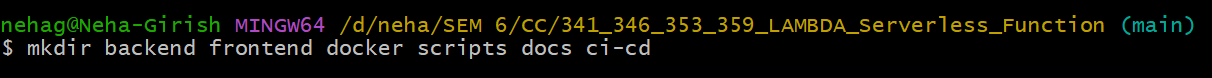


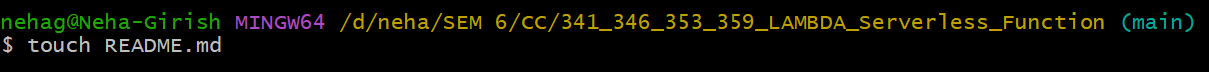
0.2: Clone and Initialize Project Structure

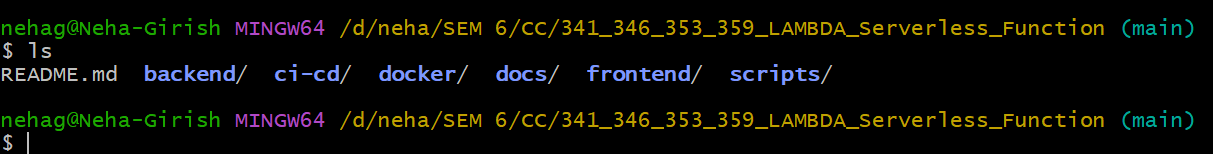
Run the following commands:



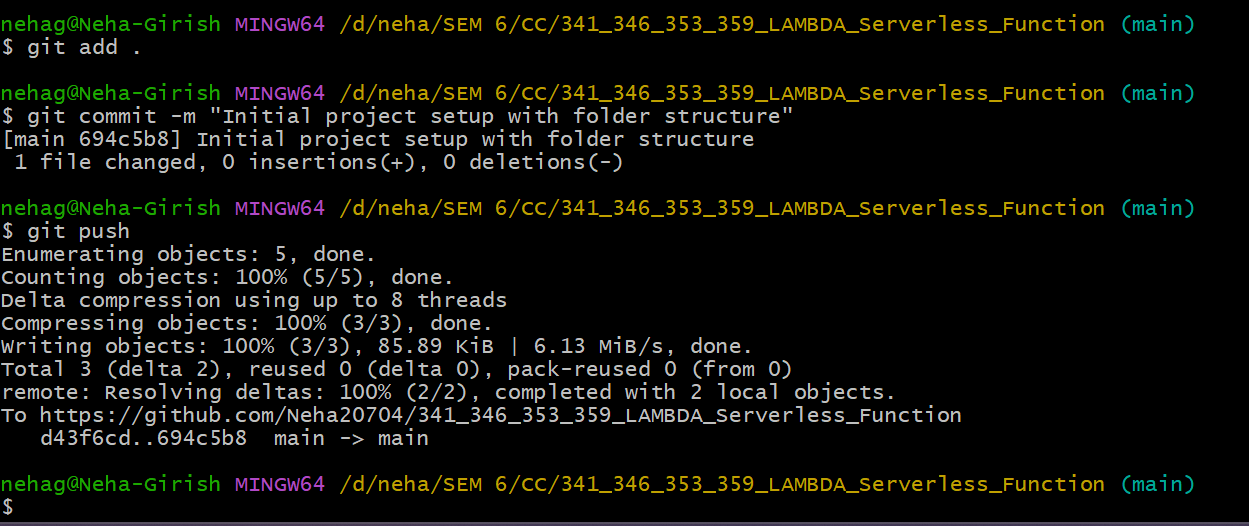




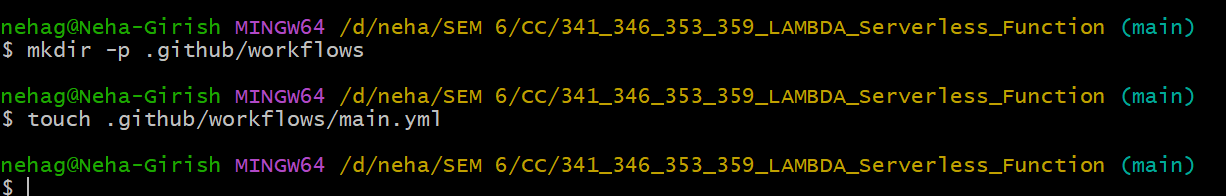


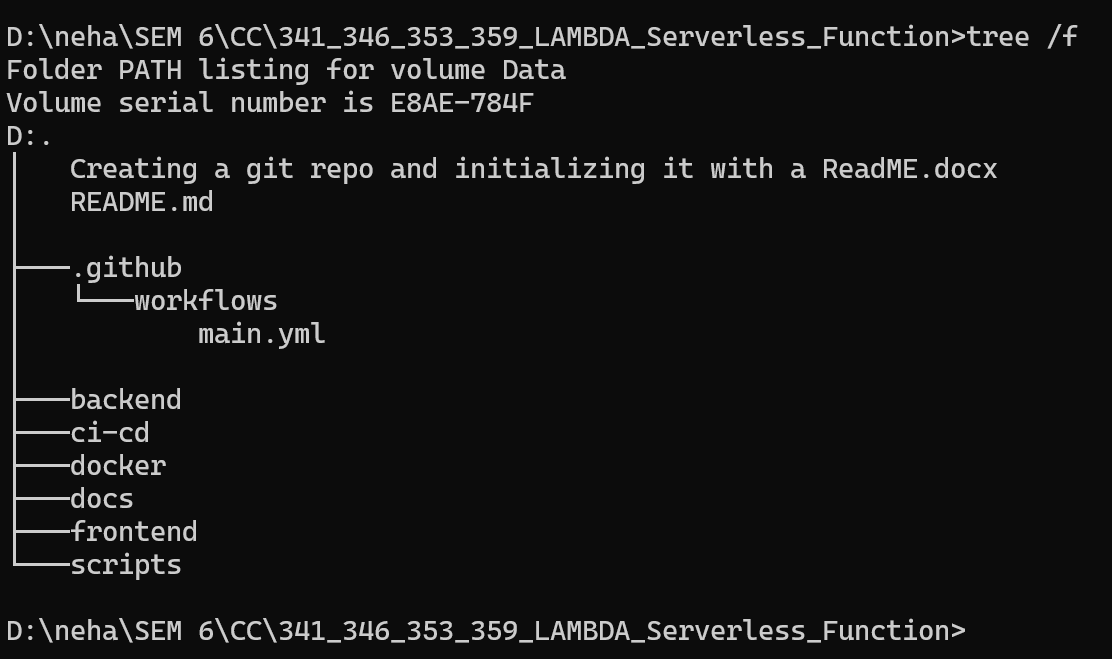


0.3: Initialize Git and Make First Commit

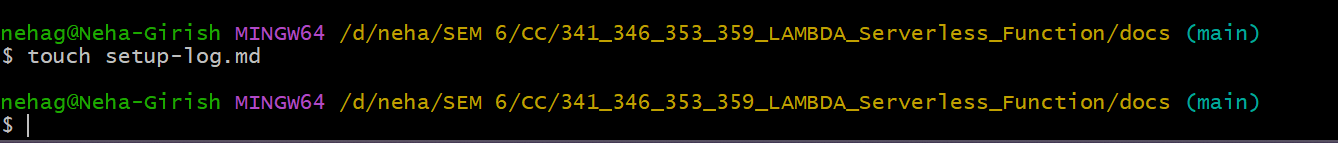


0.4: Set Up CI/CD (Optional for Now)



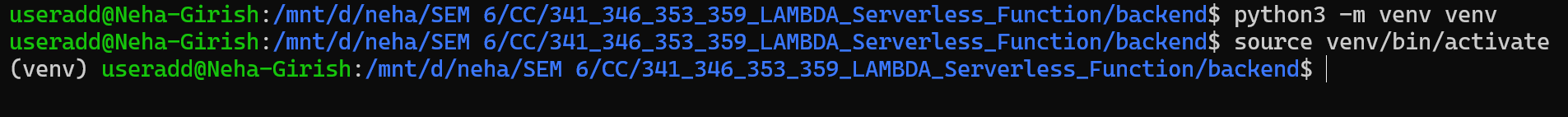


0.5: Document in /docs/setup-log.md

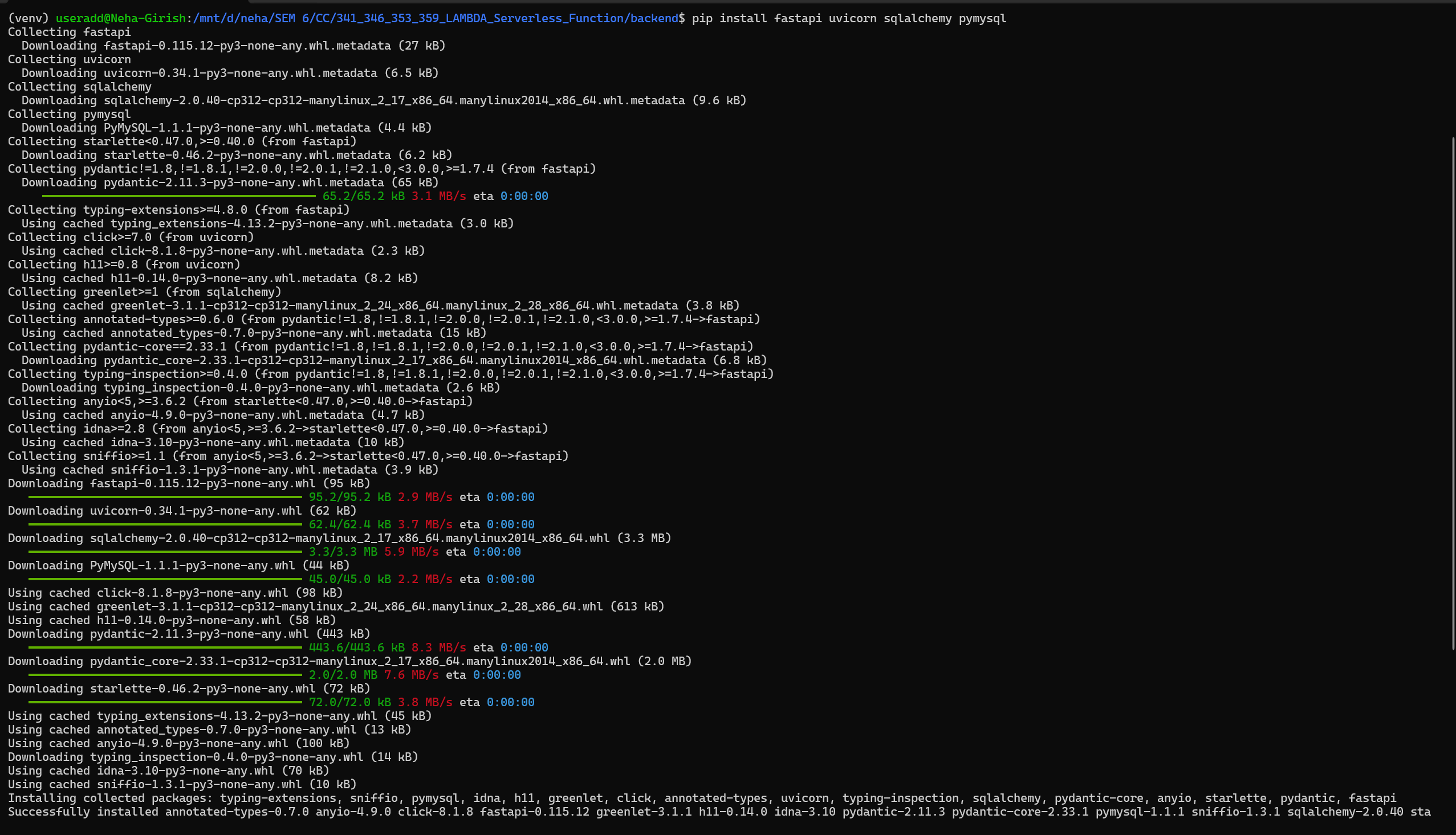


1.1: Set Up FastAPI Environment

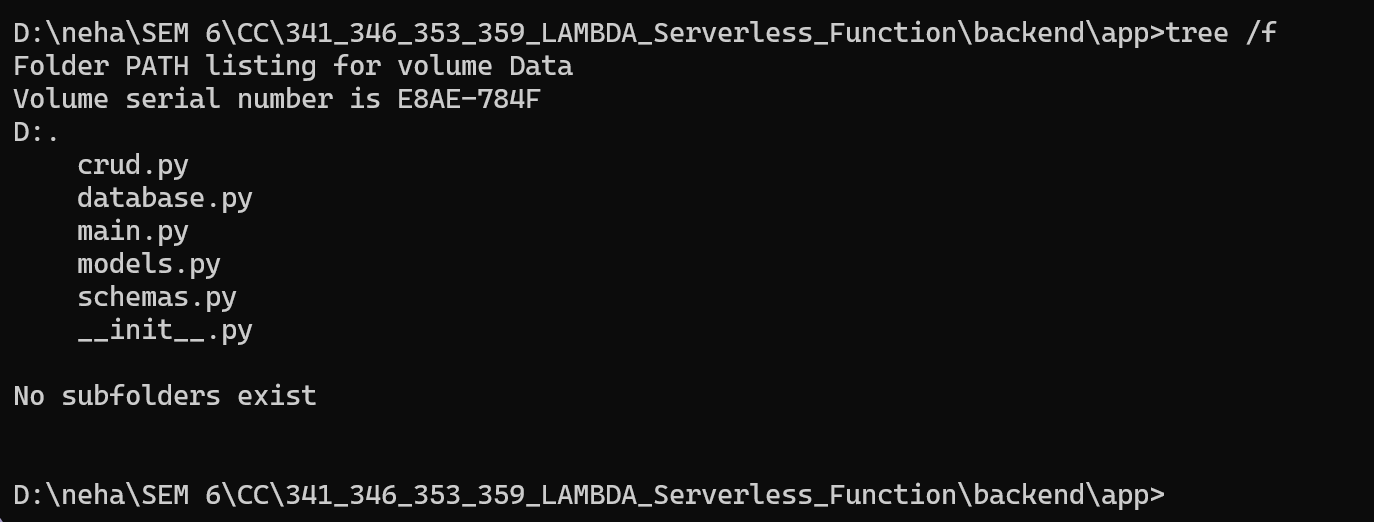
Create a Python environment:

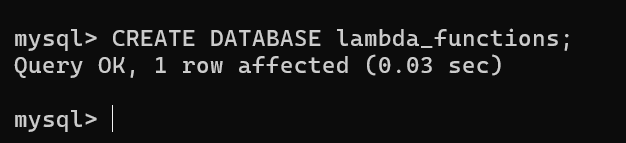


Install dependencies:



1.2: Backend Folder Structure





app/database.py

from sqlalchemy import create\_engine

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy.orm import sessionmaker

# Replace these with your actual MySQL credentials

DB\_USER = "root"

DB\_PASSWORD = "Neha@2004"

DB\_HOST = "localhost"

DB\_NAME = "lambda\_functions"

SQLALCHEMY\_DATABASE\_URL = f"mysql+pymysql://{DB\_USER}:{DB\_PASSWORD}@{DB\_HOST}/{DB\_NAME}"

engine = create\_engine(SQLALCHEMY\_DATABASE\_URL)

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

Base = declarative\_base()

app/models.py

from sqlalchemy import Column, Integer, String

from .database import Base

class Function(Base):

    \_\_tablename\_\_ = "functions"

    id = Column(Integer, primary\_key=True, index=True)

    name = Column(String(100), index=True)

    language = Column(String(50))

    route = Column(String(100), unique=True)

    timeout = Column(Integer)

app/schemas.py

from pydantic import BaseModel

class FunctionCreate(BaseModel):

    name: str

    language: str

    route: str

    timeout: int

class FunctionOut(FunctionCreate):

    id: int

    class Config:

        orm\_mode = True

app/crud.py

from sqlalchemy.orm import Session

from . import models, schemas

def get\_functions(db: Session):

    return db.query(models.Function).all()

def get\_function(db: Session, function\_id: int):

    return db.query(models.Function).filter(models.Function.id == function\_id).first()

def create\_function(db: Session, function: schemas.FunctionCreate):

    db\_func = models.Function(\*\*function.dict())

    db.add(db\_func)

    db.commit()

    db.refresh(db\_func)

    return db\_func

def delete\_function(db: Session, function\_id: int):

    func = db.query(models.Function).filter(models.Function.id == function\_id).first()

    if func:

        db.delete(func)

        db.commit()

        return True

    return False

app/main.py

from fastapi import FastAPI, Depends, HTTPException

from sqlalchemy.orm import Session

from . import models, schemas, crud, database

models.Base.metadata.create\_all(bind=database.engine)

app = FastAPI()

def get\_db():

    db = database.SessionLocal()

    try:

        yield db

    finally:

        db.close()

@app.post("/functions", response\_model=schemas.FunctionOut)

def create(function: schemas.FunctionCreate, db: Session = Depends(get\_db)):

    return crud.create\_function(db, function)

@app.get("/functions", response\_model=list[schemas.FunctionOut])

def read\_all(db: Session = Depends(get\_db)):

    return crud.get\_functions(db)

@app.get("/functions/{function\_id}", response\_model=schemas.FunctionOut)

def read\_one(function\_id: int, db: Session = Depends(get\_db)):

    func = crud.get\_function(db, function\_id)

    if not func:

        raise HTTPException(status\_code=404, detail="Function not found")

    return func

@app.delete("/functions/{function\_id}")

def delete(function\_id: int, db: Session = Depends(get\_db)):

    success = crud.delete\_function(db, function\_id)

    if not success:

        raise HTTPException(status\_code=404, detail="Function not found")

    return {"deleted": True}