#9

Data Platform Development

Lab Recap & Docker

"What do we do with missing values?
Dockerization?"

Table of Contents


Overview

Moduler:

- Pandas & Missing values
- Pydantic Field Validation
- Docker
- Dockerizing
- Hosting with Supabase (bonus)
- Environment Variables recap

Utbildningsmoment

- Dataplatfformar, bakgrund och syfte
- Git och github i teamkontext ✅
- Komponenter och teknologier i en data platform ✅
- ETL vs ELT
- Utveckling av mjukvara mot databaser ✅
- Använda Python mot relationsdatabaser och andra datakällor såsom csv, http xml/json ✅
- Använda Python mot realtidsdataströmmar såsom message queues och/eller event streaming platforms
- Använda Python och för att rensa, validera och transformera data
- Workflow processer ✅

02

Pandas Recap


Transforming Data
Missing values

analytics_summary.csv

- snittpris
- medianpris
- antal produkter
- antal produkter med saknat pris

price_analysis.csv (bonus)

- top 10 dyraste produkter
- top 10 produkter med mest avvikande pris

rejected_products.csv (bonus)

- Inkludera alla produkter som är omöjliga

På ytan ser detta väldigt enkelt ut

Men det finns ett par grå zoner och otydligheter som gör att uppgiften blir svår...

VI UTFORSKAR

Impossible Values
(Avvisa (REJECT))

Är något 'omöjligt' att bearbeta utan att förändra värdet (risk för felaktig data) så avvisas det.
Ex:

- negativa priser (tar vi bort - så blir det helt plötsligt ett positivt värde, matchar det? Omöjligt att ta reda på)
- `Created_at` datum som ligger efter `updated_at` . Omöjligt att ta reda på
- ID som är borttagna. Hur bearbetas detta? Finns det koppling till Primary_Key i databas? Om inte, omöjligt att ta reda på.

Medan överdrivna priser är prima exempel på data som oftast bör flaggas.

- Gold_plates_shoes: 7200kr (makes sense)
- bananas: 4828kr (weird - FLAG)

When do we REJECT?
(Information == Värde)

> Hur ska vi egentligen avgöra när data är “omöjlig” och ska avvisas, och när den bara är osäker och bör flaggas?
>
> Till exempel: om `currency` är NaN – är det automatiskt en reject, eller kan det finnas scenarier där vi istället flaggar?
>
> När vi flaggar något, är syftet enbart att uppmärksamma osäkerhet för manuell granskning – eller förväntas vi också korrigera värdet om det är uppenbart (t.ex. anta att det ska vara "SEK")?
>
> Var går gränsen mellan datavalidering och datakorrigering?

What if i’m wrong?
(Felaktig data & risker)

Ett annat exempel gällande felaktig data

Organisation har ett uppdrag till oss, de frågar om en kan analysera top 10 månads intäckter -> vi glömmer att städa data -> och median fallerar för att vi råkat ha felaktig data med -> dåligt resultat -> reflekteras dåligt på oss som utvecklare

What if i’m wrong? #2
(Felaktig data & risker)

Precis!
För nu handlar det delvist om konsekvenstänkande.
Låt oss säga att vi släpper in

```
Sek 500
```

```
currency NONE
```

Om vi inte kan få reda på informationen så är det **direkt avvisande**.
Är det USD? EUR? SEK? Detta kan bli väldigt felaktig data i längden om vi stöter på fler fall. Denna grå zon är perfekt exempel på när det är viktigt att analysera organisationens behov.

I vårt fall - **avvisa direkt**

Datakvalitet kan bli affärsrisk.
Vi får aldrig gissa affärskritisk information i ett datalager.
Antaganden hör hemma i kravspecifikation (i detta fall har vi inget, därav blir det svårt, vilket är avsiktligt för denna labboration)

Här kommer ett exempel:

--PRICE LIST--

```
price
------
100
200
free
300
```

Om vi sedan kalkylerar .mean() på 'none

Python

```python
prices = pd.Series([100, 200, None, 300])
prices.mean()
```

resultat på medelvärdet blir **200**. *((100 + 200 + 300) / 3 = 200)*
Varför är detta farligt?

Solution
(Felaktig data & risker)

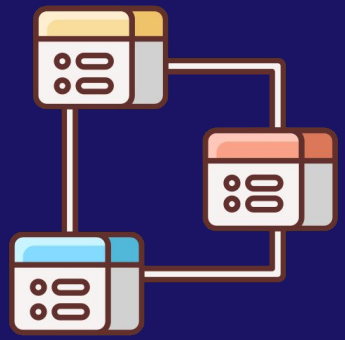
Precis!
Svaret blir ett annat resultat: (100 + 200 + 0 + 300) / 4 = 150

Om data inte går att använda
Är datan omöjlig att bearbeta, omöjlig att gissa sig fram till = avvisa

None & NaN
Betyder att värdet är okänt. Om resterande av raden är giltig och innehåller värdefull information = flagga

Free
Betyder att värdet är känt, men inte numerisk representation

Data modelling?
(Felaktig data & risker)



I ett verkligt scenario *(överkurs just nu)*
Innan du ens tittar på `NaN` måste du veta:

- Är pris obligatoriskt?
- Får pris vara 0?
- Får pris vara negativt?
- Får pris saknas?

Detta är ett **datakontrakt relaterad fråga**, det är därför det blir förvirrande då vi arbetar inom pandas för att försöka lösa problemet.
Det är här ***datamodellering*** kommer in

What if i’m wrong? #3
(Felaktig data & risker)

Ett sista exempel - Datamodellering

Datamodellering löser oftast problemet med osäkerhet om vad som får komma in genom att definiera om datan är obligatoriskt tillåten eller meningsfull

```
price_amount   DECIMAL NOT NULL
currency_code  CHAR(3) NOT NULL
```

KPI vs Quality Assurance
(The Difference)

Key Performance Indicator

Mäter hur bra verksamheten presterar

- Genomsnittligt ordervärde
- Antal aktiva användare
- Andel godkända transaktioner
- Omsättning per månad

Quality Assurance

Säkerställer att datan är korrekt, komplett och tillförlitlig

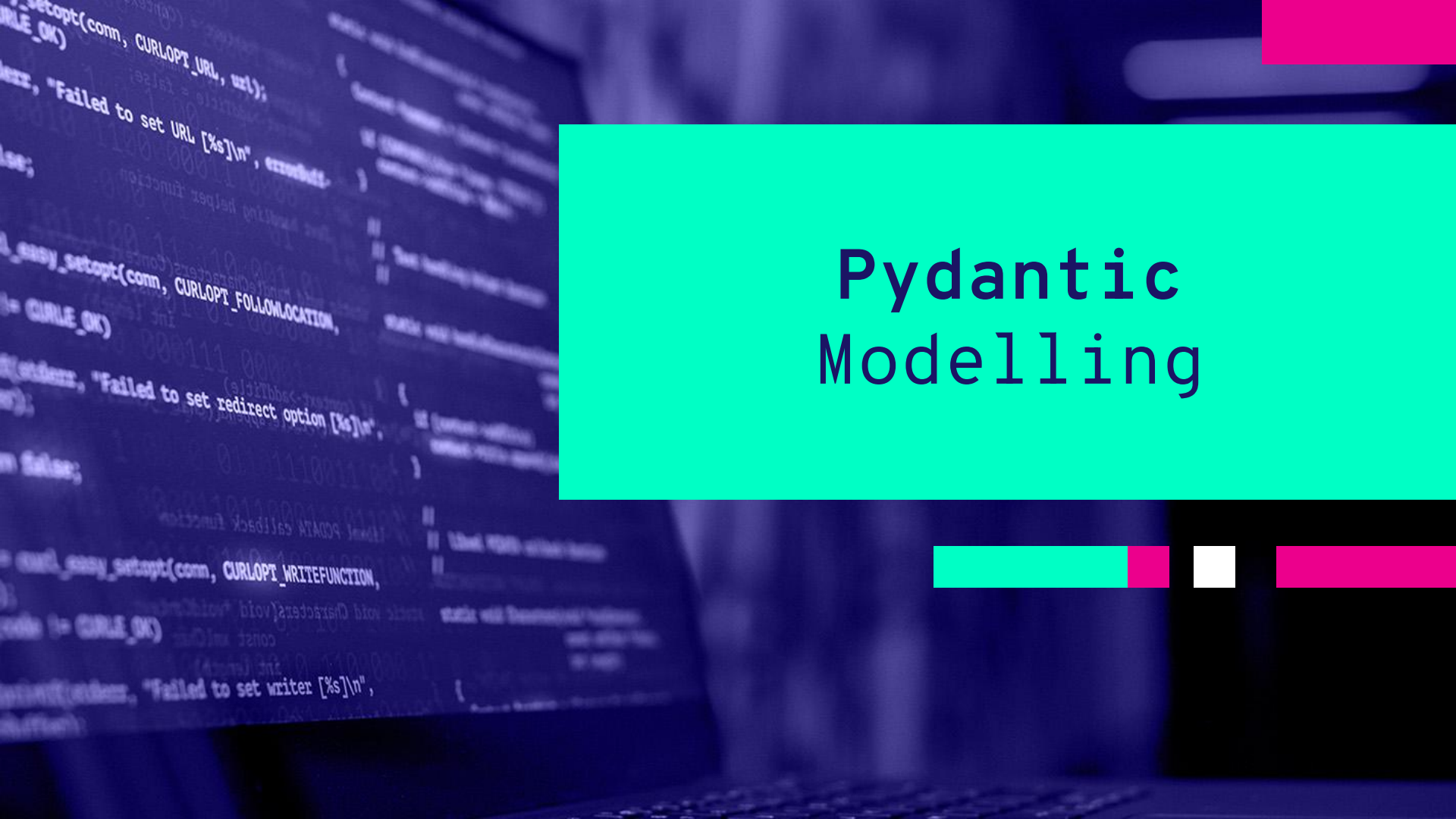
- Saknade värden (NaN)
- Negativa priser
- Felaktigt datumformat
- Dubbletter

*Kvalitetssäkring är **förutsättningen** för att KPI:er ska vara trovärdiga.*

Summarized
(Information == Värde)

Finansiell rapportering / KPI:er
oftast endast *valid data*

Explorativ analys / kvalitetssäkring
valid + flaggad (med transparens)


Pydantic
Modelling

```python
from pydantic import BaseModel, Field


class Product(BaseModel):
    name: str = Field(
        min_length=2,
        max_length=100,
        description="Product name between 2 and 100 characters"
    )

    price: float = Field(
        gt=0,
        le=10000,
        description="Price must be greater than 0 and max 10,000"
    )

    quantity: int = Field(
        ge=0,
        le=1000,
        description="Stock cannot be negative and max 1000"
    )

    sku: str = Field(
        pattern=r"^SKU-\d{4}$",
        description="Format must be SKU-1234"
    )
```

03

Docker

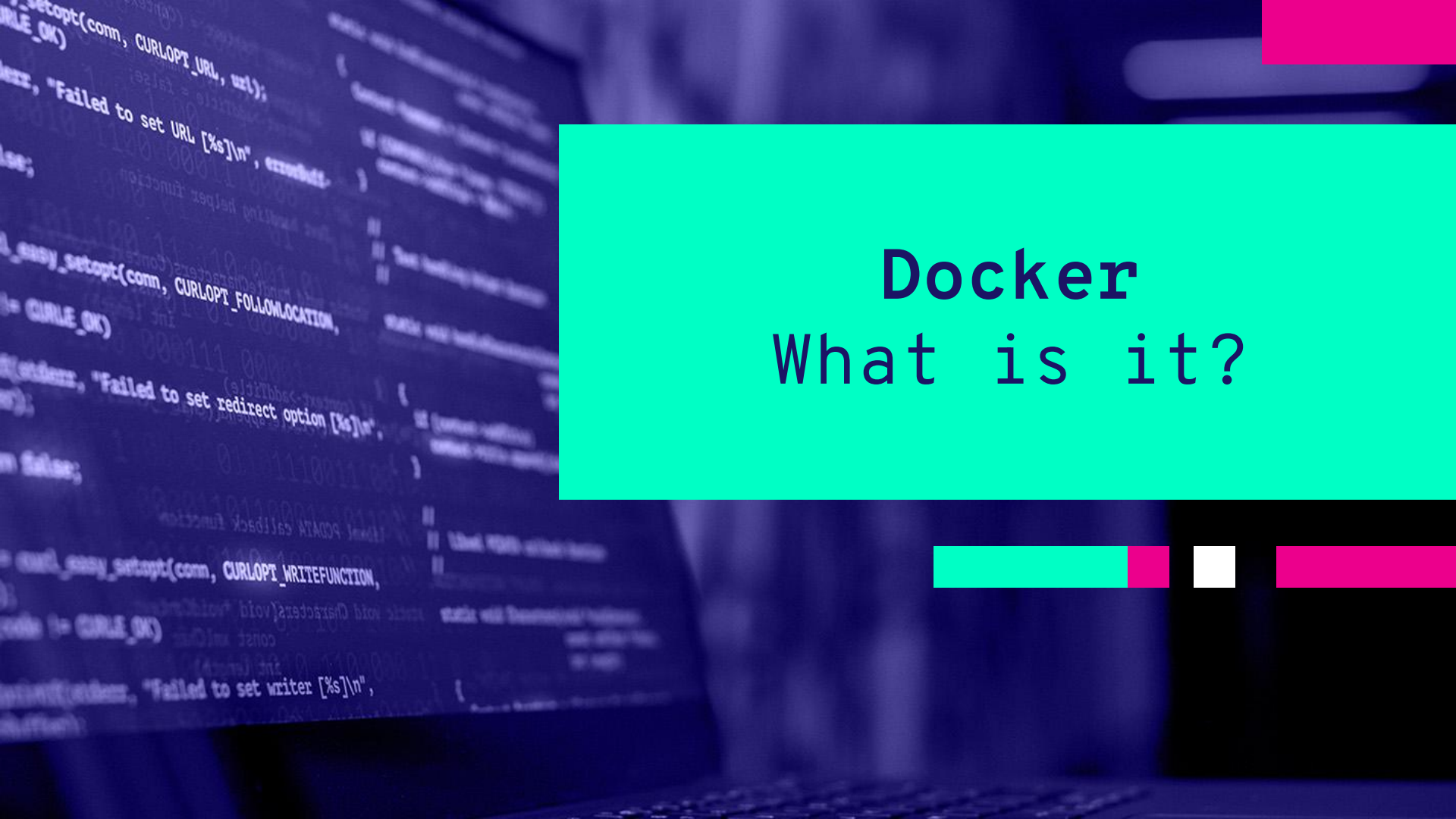
Docker
Install & Version

Is docker installed?
(command)

Installation
(MAC & Windows)

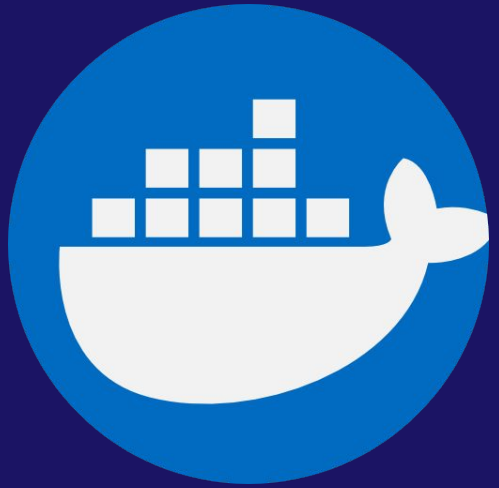
MAC
https://docs.docker.com/desktop/setup/install/mac-install/

Windows
https://docs.docker.com/desktop/setup/install/windows-install/


Docker
What is it?

Docker
(What is it)

Docker
An **open source platform** that allows developers and systems administrators to **package applications** into **containers**



Docker
(What is it)

What is Docker?

Copy as Markdown

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code, you can significantly reduce the delay between writing code and running it in production.

Read more: https://docs.docker.com/get-started/docker-overview/

Docker Terminology
Container

The Problem

Explanation

Imagine you're developing a killer web app that has three main components - a React frontend, a Python API, and a PostgreSQL database. If you wanted to work on this project, you'd have to install Node, Python, and PostgreSQL.

How do you make sure you have the same versions as the other developers on your team? Or your CI/CD system? Or what's used in production?

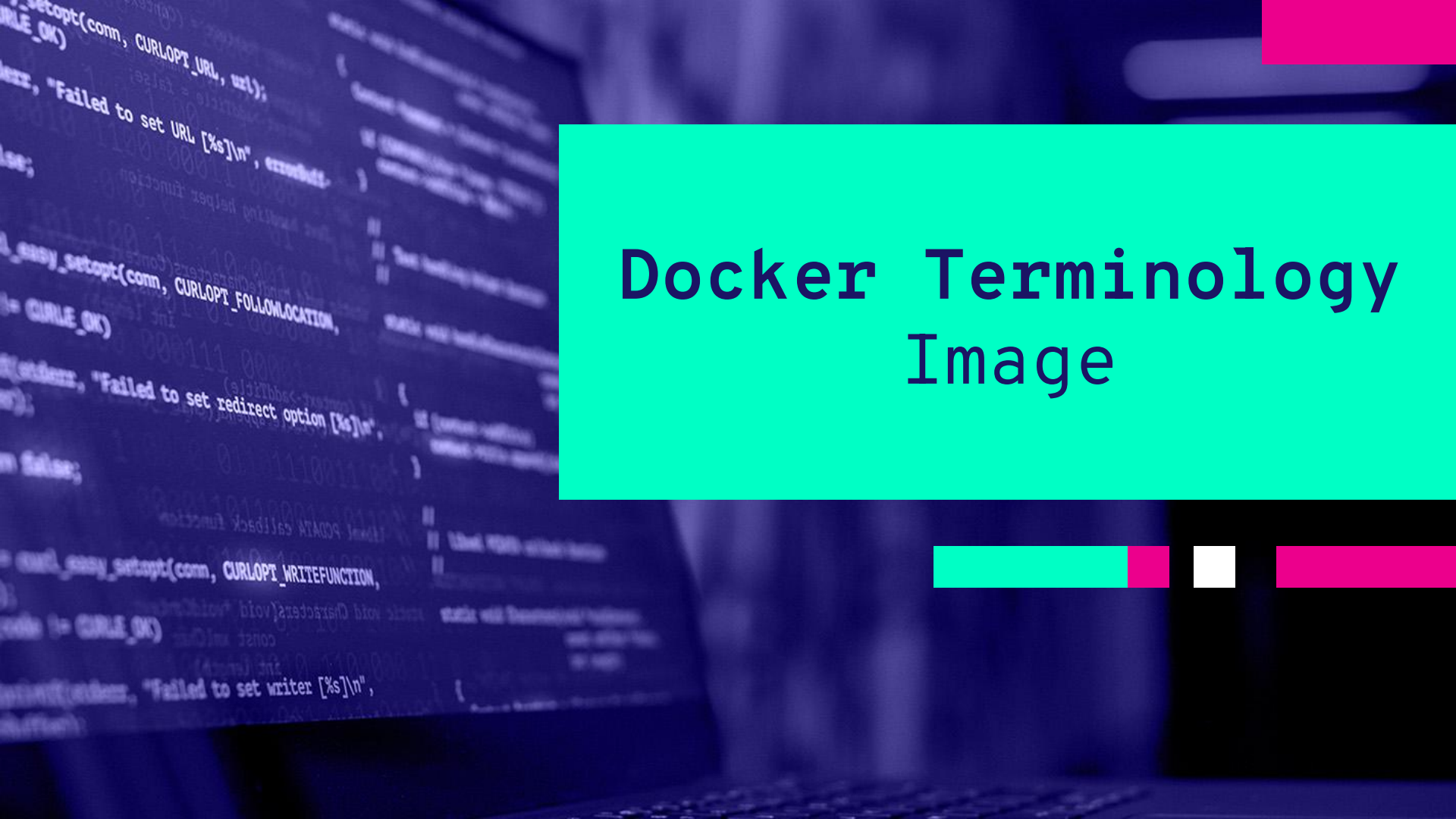
How do you ensure the version of Python (or Node or the database) your app needs isn't affected by what's already on your machine? How do you manage potential conflicts?

Source: https://docs.docker.com/get-started/docker-concepts/the-basics/what-is-a-container

What is a container? Simply put, containers are isolated processes for each of your app's components. Each component - the frontend React app, the Python API engine, and the database - runs in its own isolated environment, completely isolated from everything else on your machine.

Here's what makes them awesome. Containers are:

- Self-contained. Each container has everything it needs to function with no reliance on any pre-installed dependencies on the host machine.
- Isolated. Since containers run in isolation, they have minimal influence on the host and other containers, increasing the security of your applications.
- Independent. Each container is independently managed. Deleting one container won't affect any others.
- Portable. Containers can run anywhere! The container that runs on your development machine will work the same way in a data center or anywhere in the cloud!


Docker Terminology
Image

Image

Explanation

Seeing as a container is an isolated process, where does it get its files and configuration? How do you share those environments?

That's where container images come in. A container image is a standardized package that includes all of the files, binaries, libraries, and configurations to run a container.

For a PostgreSQL image, that image will package the database binaries, config files, and other dependencies. For a Python web app, it'll include the Python runtime, your app code, and all of its dependencies.

Source: https://docs.docker.com/get-started/docker-concepts/the-basics/what-is-an-image/

Docker Terminology
Dockerfile

Dockerfile

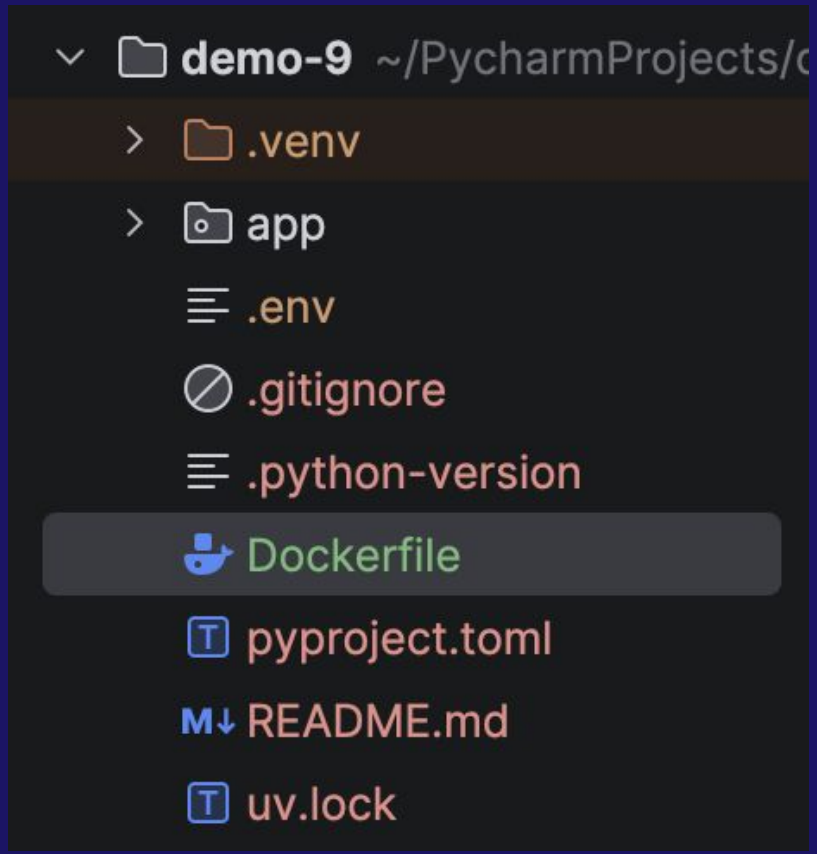
Dockerfile reference

Copy as Markdown

Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. This page describes the commands you can use in a Dockerfile.

Source: https://docs.docker.com/reference/dockerfile/

Dockerfile Example



Den MÅSTE döpas till ‘Dockerfile’

Dockerfile Example #2

```
FROM python:3.12-slim

WORKDIR /app

# Install uv
RUN pip install uv

# Copy only dependency files first (better layer caching)
COPY pyproject.toml uv.lock ./

# Install dependencies from lock file
RUN uv sync --frozen

# Copy rest of the project
COPY . .

EXPOSE 8000

CMD ["uv", "run", "uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Dockerfile Instructions

`ONBUILD`	Specify instructions for when the image is used in a build.
`RUN`	Execute build commands.
`SHELL`	Set the default shell of an image.
`STOPSIGNAL`	Specify the system call signal for exiting a container.
`USER`	Set user and group ID.
`VOLUME`	Create volume mounts.
`WORKDIR`	Change working directory.

Instruction	Description
`ADD`	Add local or remote files and directories.
`ARG`	Use build-time variables.
`CMD`	Specify default commands.
`COPY`	Copy files and directories.
`ENTRYPOINT`	Specify default executable.
`ENV`	Set environment variables.
`EXPOSE`	Describe which ports your application is listening on.
`FROM`	Create a new build stage from a base image.
`HEALTHCHECK`	Check a container's health on startup.
`LABEL`	Add metadata to an image.
`MAINTAINER`	Specify the author of an image.

Docker Commands

Common

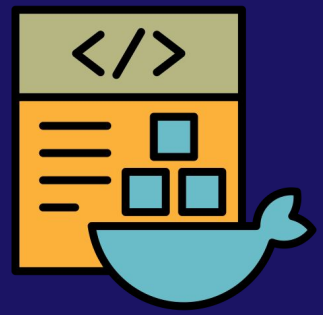
Docker Commands
(List running containers)

```
$ docker ps
```

Lists all docker containers.
Add -a at the end to list stopped ones.

Docker Commands
(Generate an Image)

```
$ docker build -t demo-9 .
```



-t == tag name

. == from where we currently stand (path)

Docker Commands
(Generate an Image)

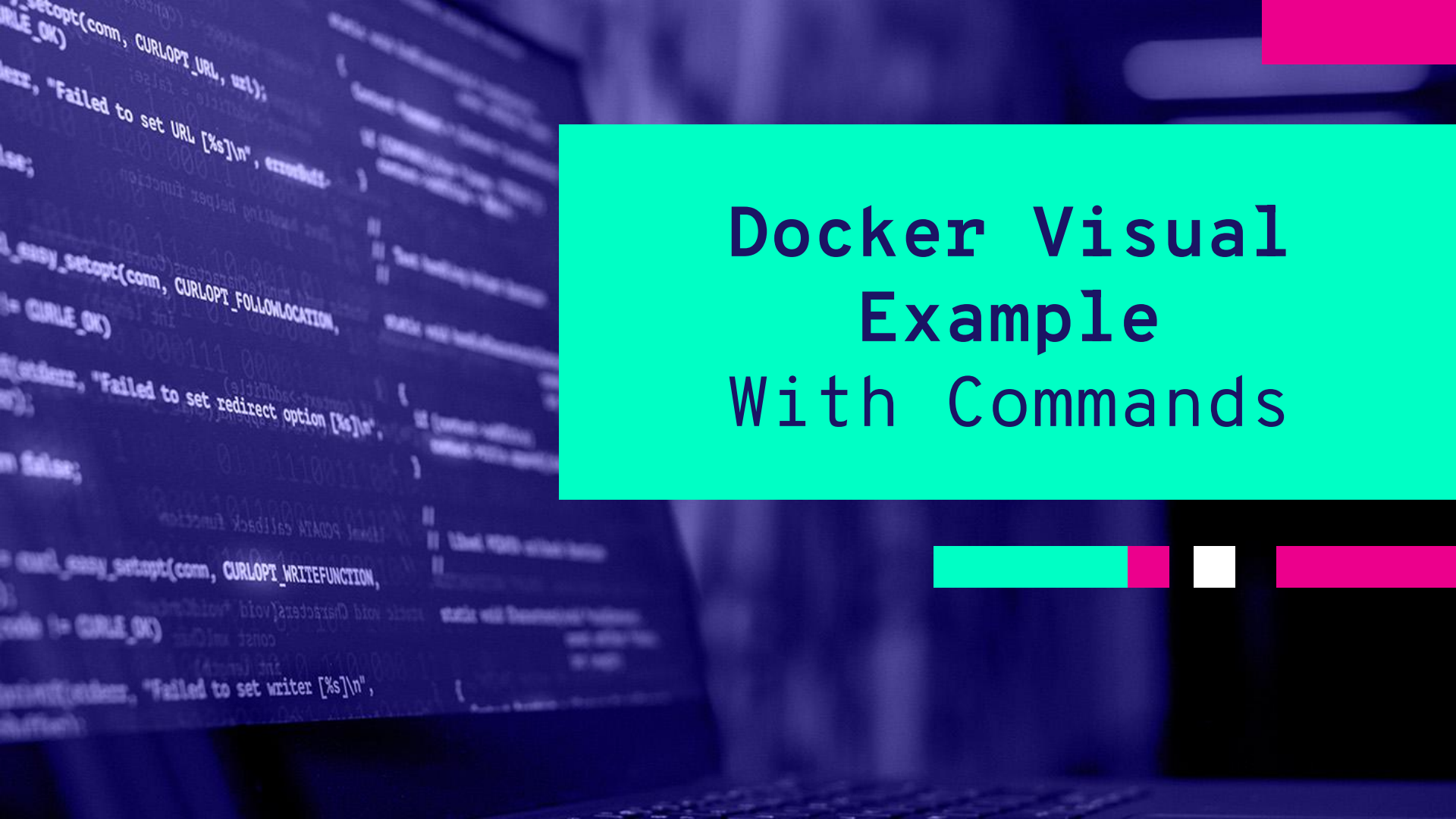
```
$ docker run -p 8000:8000 demo-9
```

Create a container with an Image inside
Expose port 8000 to the local machine
Demo-9 is the image in question, without it this command won't work!
Rekommendera att testa utan -p 8000:8000 för debugging

Docker Commands BONUS
(Generate an Image)

```
$ docker run --rm -p 8000:8000
demo-9
```

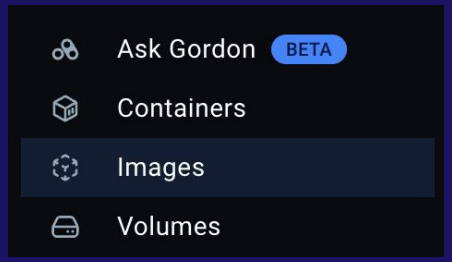
Docker Hygiene by removing old container


Docker Visual
Example
With Commands

Build Image
(docker build -t NAME .)

This means success!

```
 => => extracting sha256:09fa645f2e8f01bf76e8432e1b0c11b79a8726f64b60501af160c71b7c3917ab
 => [internal] load build context
 => => transferring context: 53.13MB
 => [2/6] WORKDIR /app
 => [3/6] RUN pip install uv
 => [4/6] COPY pyproject.toml uv.lock ./
 => [5/6] RUN uv sync --frozen
 => [6/6] COPY . .
 => exporting to image
 => => exporting layers
 => => writing image sha256:a215570c97a1ea19f258c9b3fa13eb2392f960dd3fc38c1286e01d259a175778
 => => naming to docker.io/library/demo-9

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/u7xr3e0ivmx8rwy4yt4aiy5kd
```



Local My Hub

0 Bytes / 765.74 MB in use 3 images

Last refresh: 45 minutes ago

Search

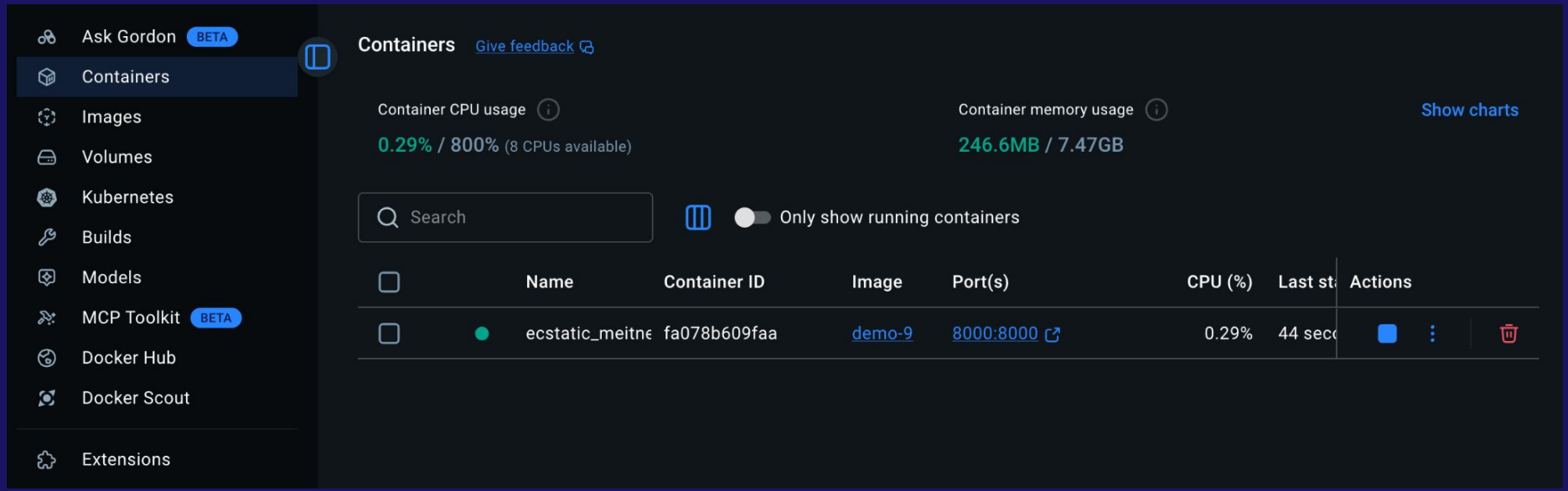
Name	Tag	Image ID	Created	Size	Actions
postgres	latest	21a7e541217c	5 months ago	478.92 MB	
testcontainers/ryuk	0.12.0	3b9e208a7ab3	9 months ago	15.84 MB	
demo-9	latest	a215570c97a1	2 minutes ago	350.27 MB	

Run App
(docker run demo-9)

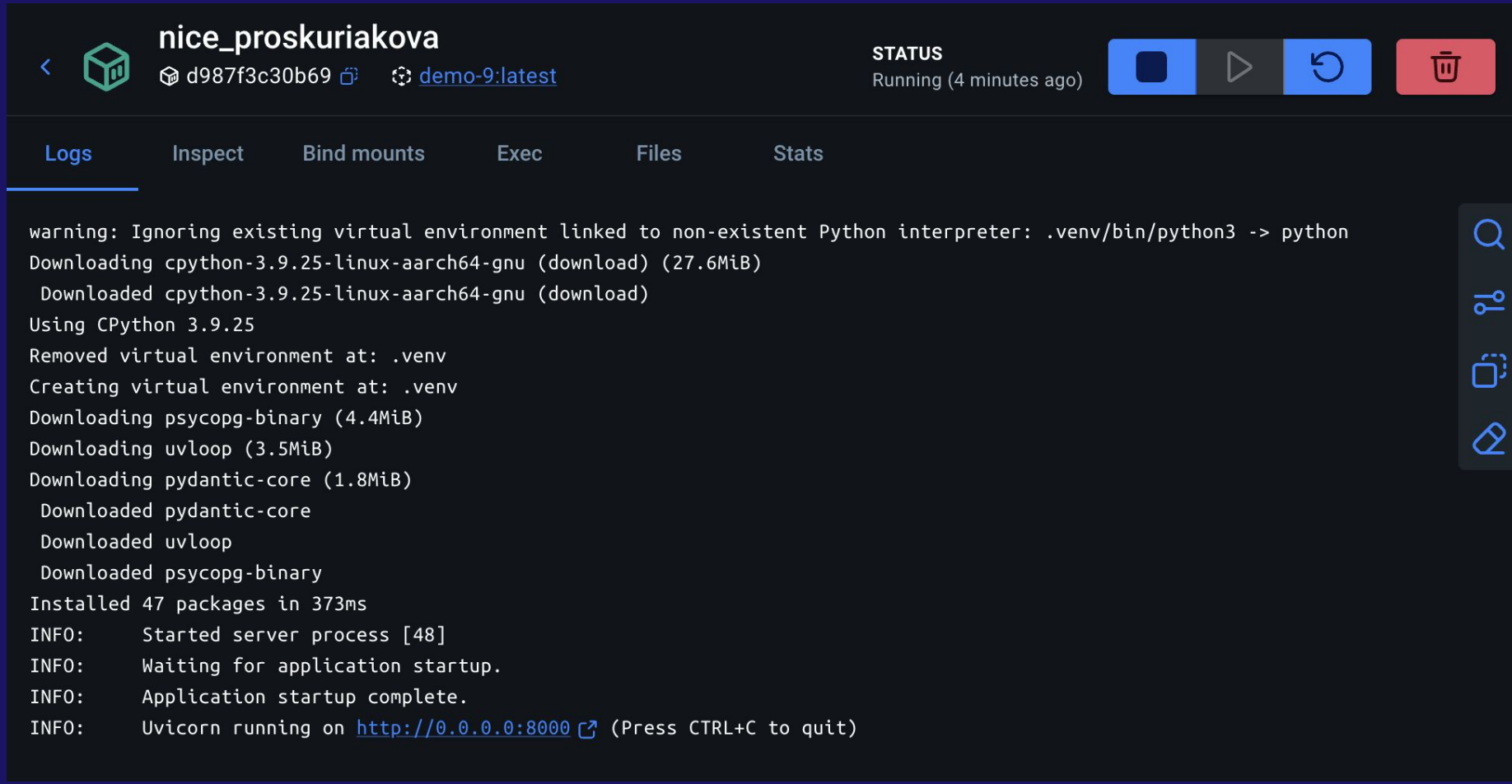
		Name	Tag	Image ID	Created	Size	Actions
☐	○	postgres	latest	21a7e541217c	5 months ago	478.92 MB	▷ ⋮ 🗑
☐	○	testcontainers/ryuk	0.12.0	3b9e208a7ab3	9 months ago	15.84 MB	▷ ⋮ 🗑
☐	●	demo-9	latest	a215570c97a1	15 minutes ag	350.27 MB	▷ ⋮ 🗑

Notice it's green!

Container up and running? (YES)



Notice it has exposed port 8000!


nice_proskuriakova
d987f3c30b69
demo-9:latest
STATUS
Running (4 minutes ago)
Logs
Inspect
Bind mounts
Exec
Files
Stats
warning: Ignoring existing virtual environment linked to non-existent Python interpreter: .venv/bin/python3 -> python
Downloading cpython-3.9.25-linux-aarch64-gnu (download) (27.6MiB)
Downloaded cpython-3.9.25-linux-aarch64-gnu (download)
Using CPython 3.9.25
Removed virtual environment at: .venv
Creating virtual environment at: .venv
Downloading psycopg-binary (4.4MiB)
Downloading uvloop (3.5MiB)
Downloading pydantic-core (1.8MiB)
Downloaded pydantic-core
Downloaded uvloop
Downloaded psycopg-binary
Installed 47 packages in 373ms
INFO: Started server process [48]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)


So… What's the big deal?
Explained

Container up and running? (YES)

Reproducible environment
Same Python, same OS, same dependencies… everywhere!

No “works on my machine” problems
What runs locally runs the same in production.

Isolation
No dependency conflicts, no global pollution, easy cleanup.

Deployable unit
The app + runtime + config are packaged into one portable artifact.

04

Hosting (bonus) & Docker Env variables


.env
FIX


Hosting
PostgreSQL

Build in a weekend
Scale to millions

Supabase is the Postgres development platform.
Start your project with a Postgres database, Authentication, instant APIs, Edge Functions, Realtime subscriptions, Storage, and Vector embeddings.

Supabase
(Free Tier)

$0 / month

Get started with:

- ✓ Unlimited API requests
- ✓ 50,000 monthly active users
- ✓ 500MB database size
  Shared CPU • 500MB RAM
- ✓ 5 GB egress
- ✓ 5 GB cached egress
- ✓ 1 GB file storage
- ✓ Community support

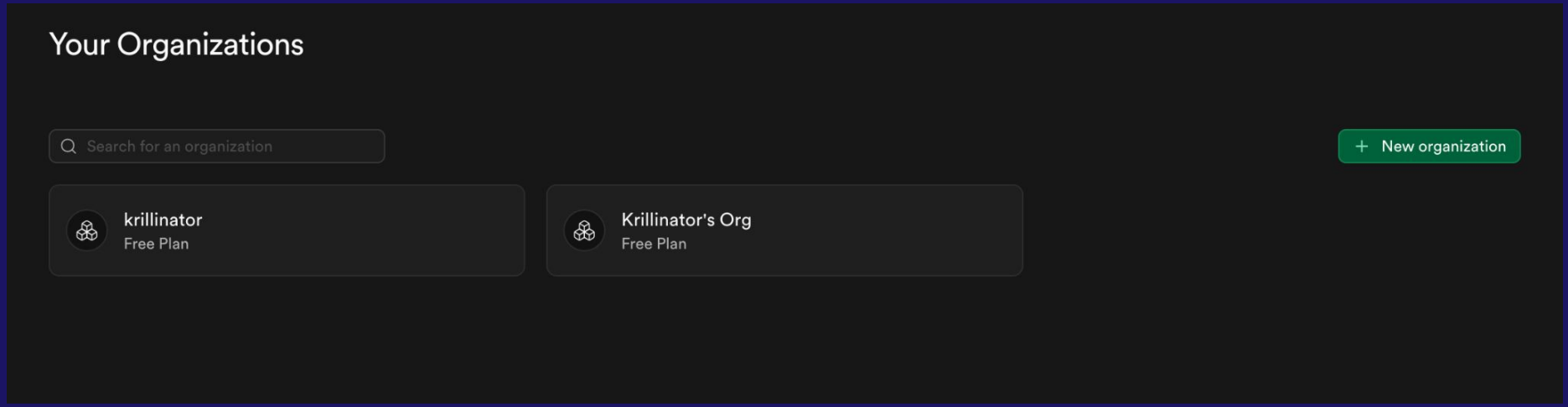
Read more: https://supabase.com/pricing

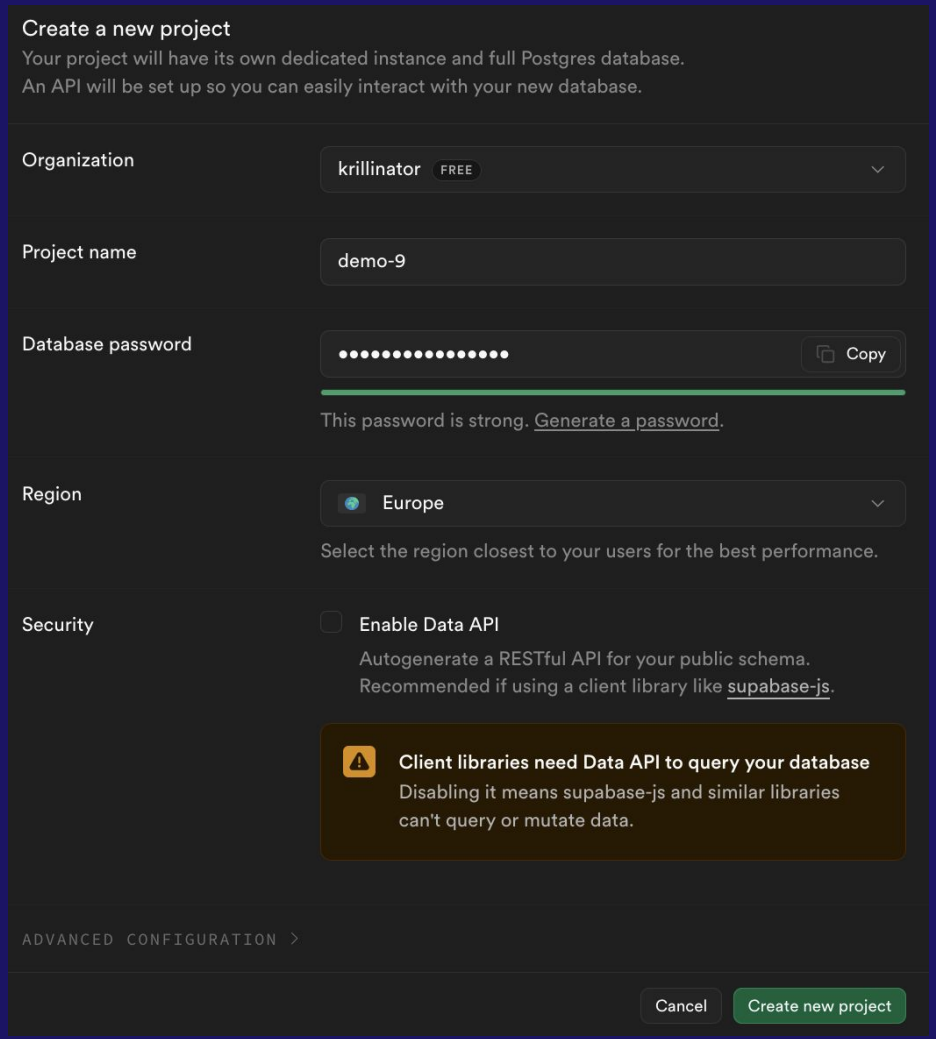
GitHub + Supabase
(Connect to Account)

https://supabase.com/

Sign up using GitHub for easy repository access

Create Organization
(HUB for projects)



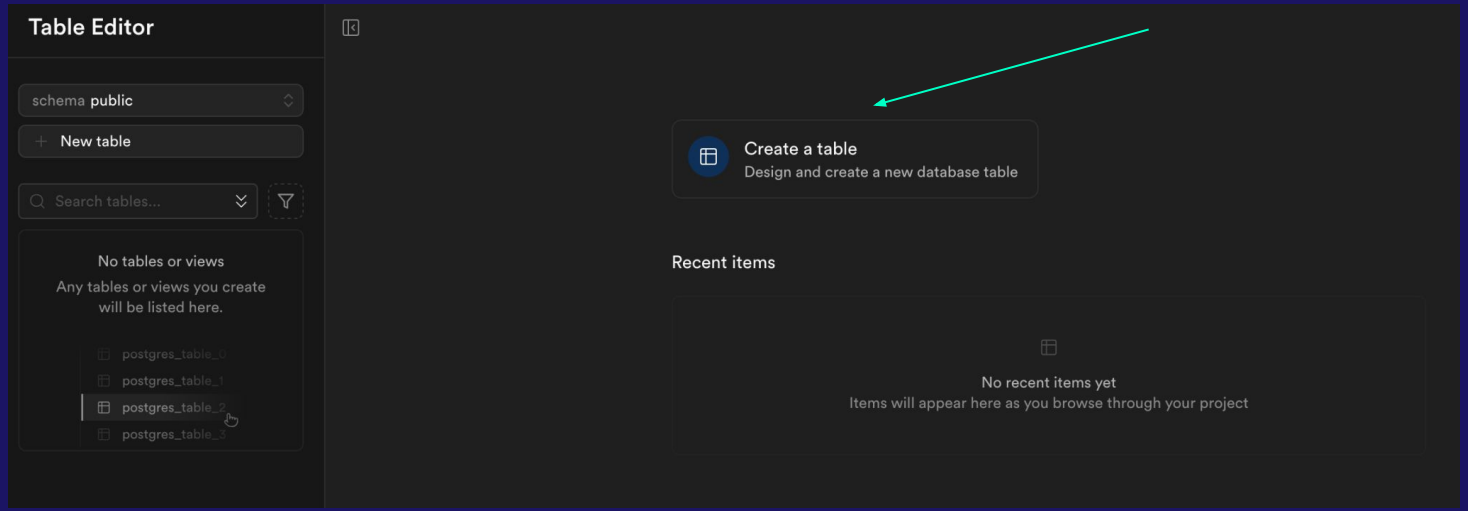


New Project #2 (Disable Data API)

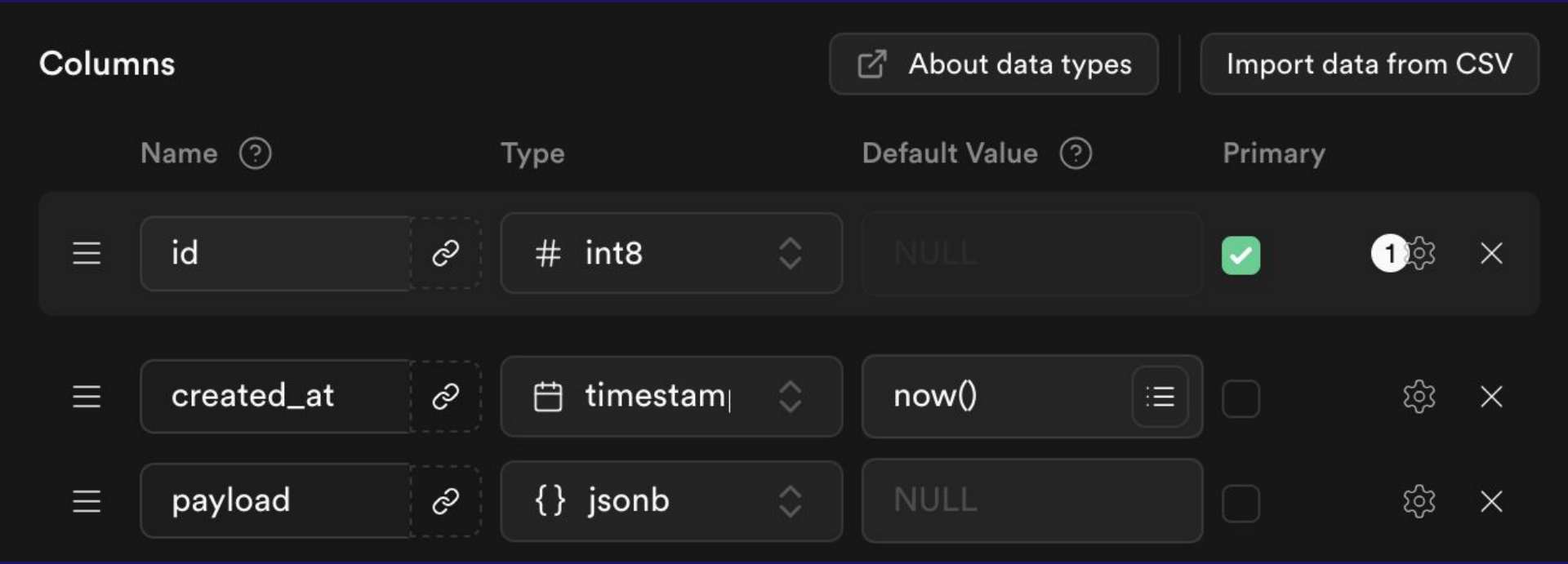
We disable it because we don't want to use it against a frontend UI.

We're using **psycopg3** with Python instead

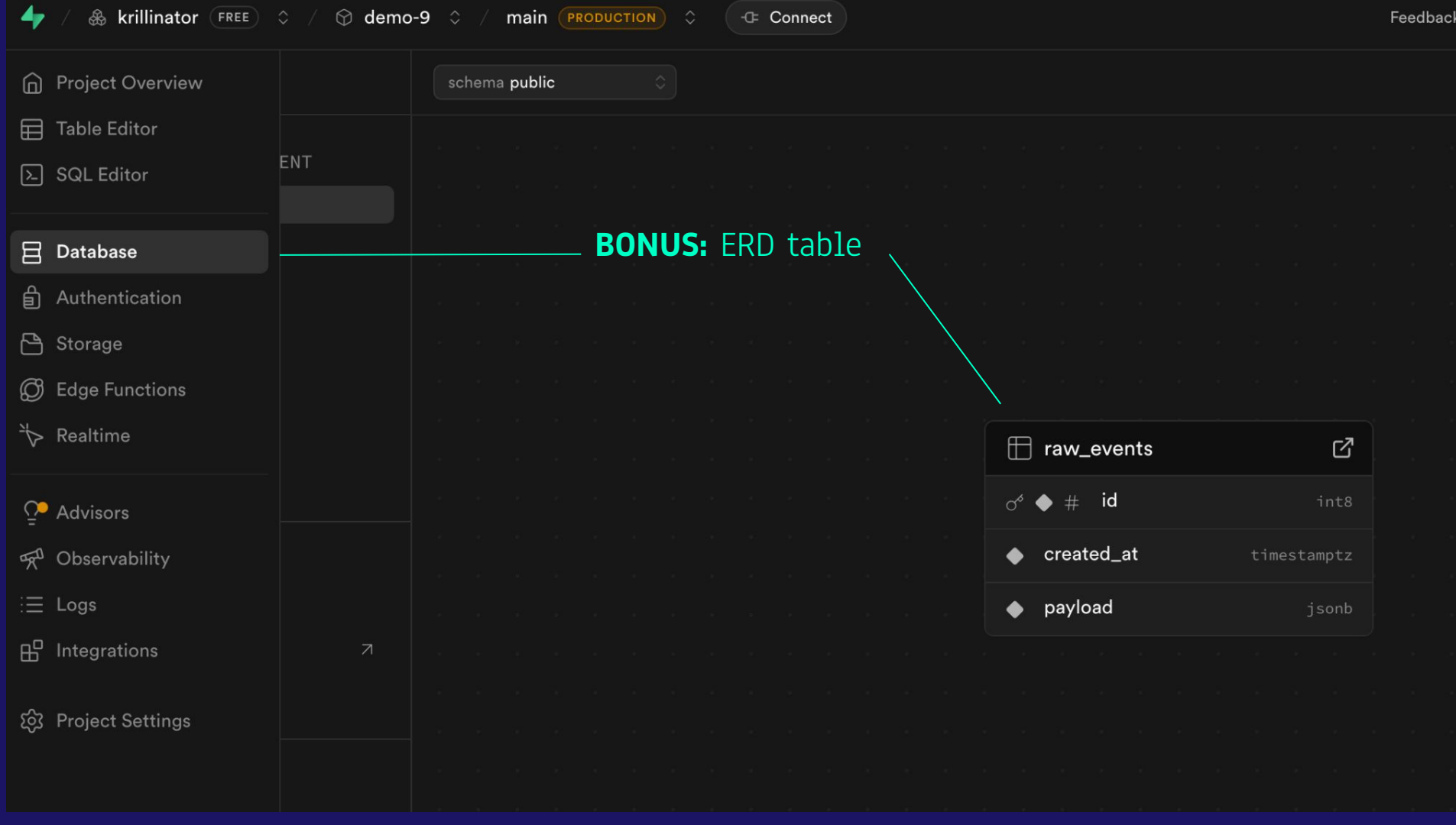
Create First Table
(Supabase)



Create First Table #2
(Payload: jsonB)

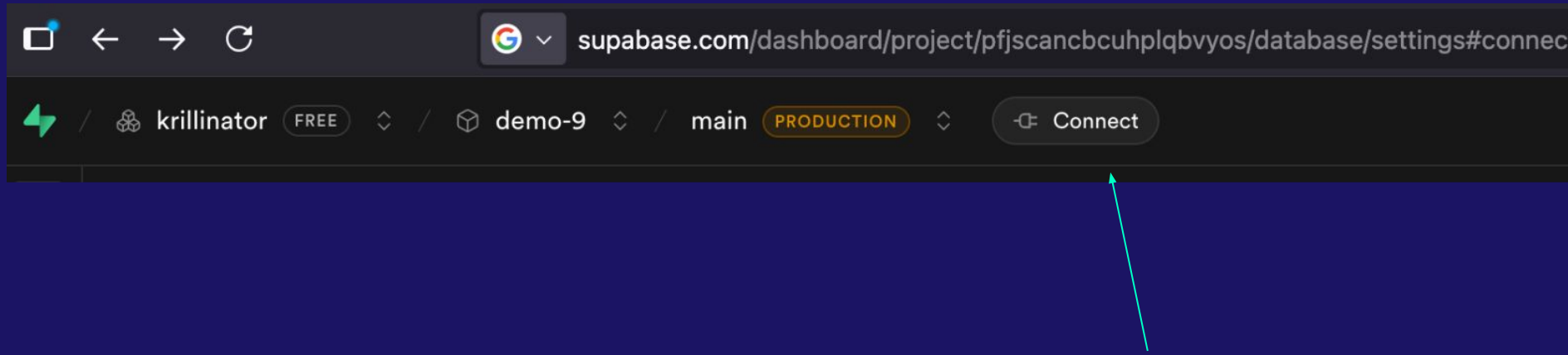


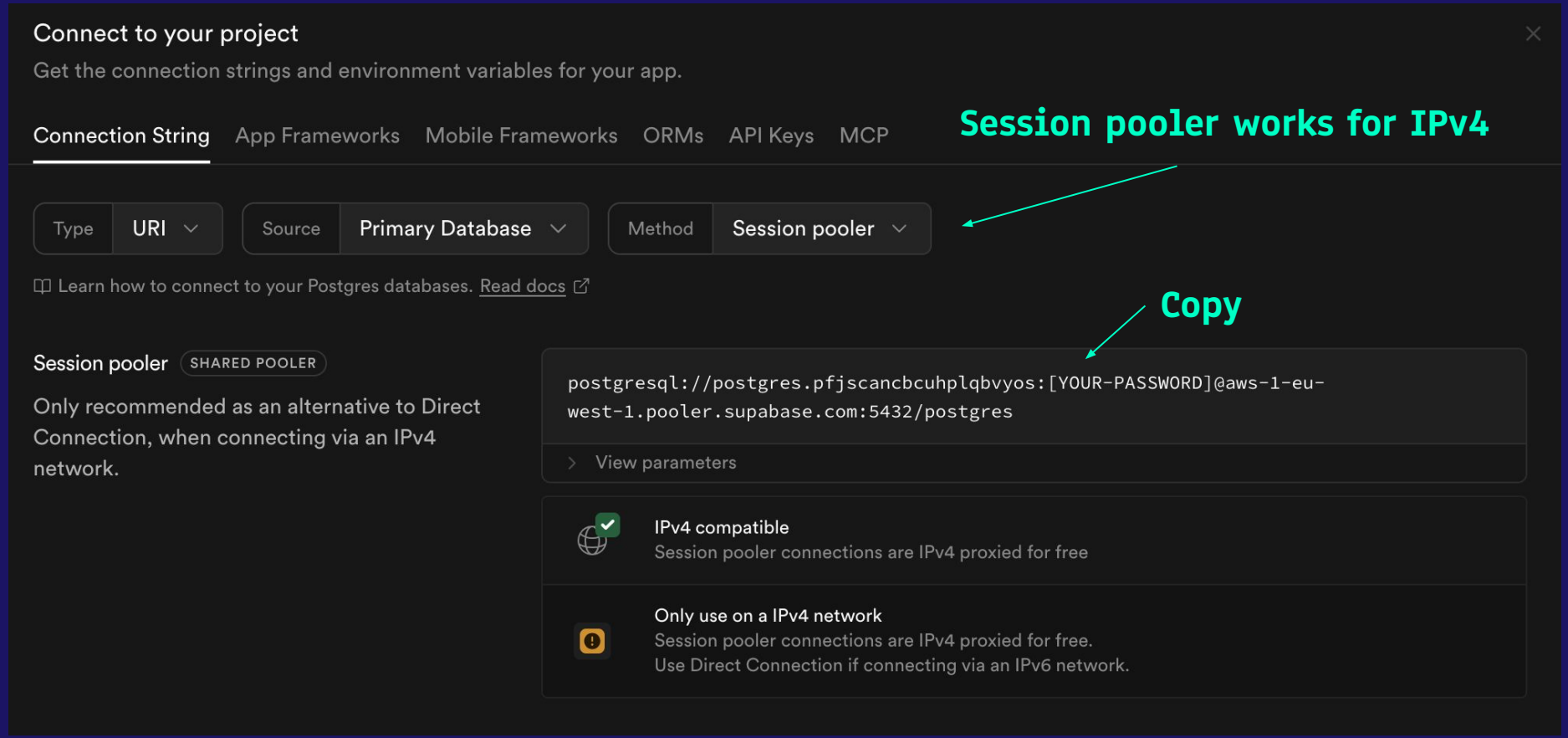
Notice: You can also import data from CSV!


krillinator
FREE
demo-9
main
PRODUCTION
Connect
Feedback
Project Overview
Table Editor
SQL Editor
Database
Authentication
Storage
Edge Functions
Realtime
Advisors
Observability
Logs
Integrations
Project Settings
schema public
ENT
BONUS: ERD table
raw_events
id
int8
created_at
timestamptz
payload
jsonb


Connection String
IPv4

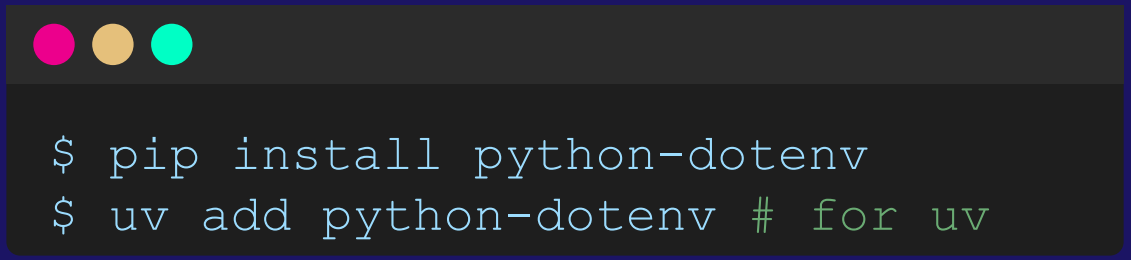
Create First Table
(Supabase)



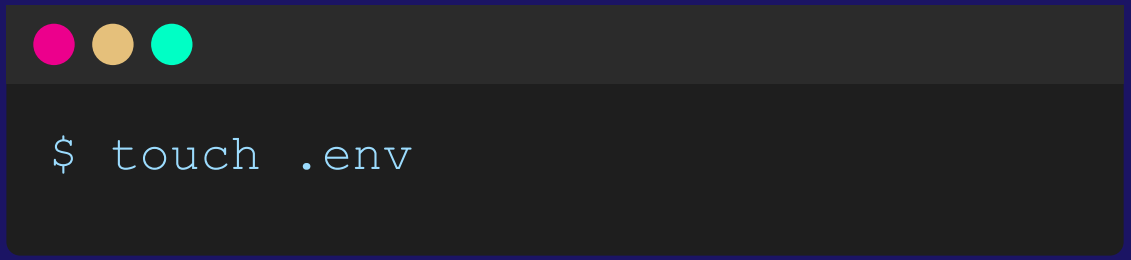

Connect to your project
Get the connection strings and environment variables for your app.
Connection String
App Frameworks
Mobile Frameworks
ORMs
API Keys
MCP
Session pooler works for IPv4
Type
URI
Source
Primary Database
Method
Session pooler
Learn how to connect to your Postgres databases. Read docs
Copy
Session pooler
SHARED POOLER
Only recommended as an alternative to Direct Connection, when connecting via an IPv4 network.
postgresql://postgres.pfjscancbcuhplqbvyos:[YOUR-PASSWORD]@aws-1-eu-west-1.pooler.supabase.com:5432/postgres
View parameters
IPv4 compatible
Session pooler connections are IPv4 proxied for free
Only use on a IPv4 network
Session pooler connections are IPv4 proxied for free.
Use Direct Connection if connecting via an IPv6 network.


Environment
Variables
Step by Step

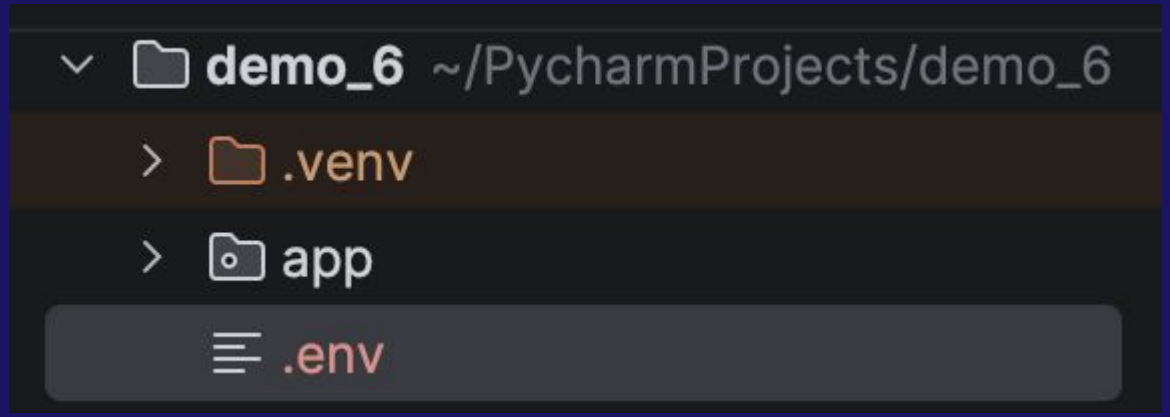
Environment Variables
(Dependency)



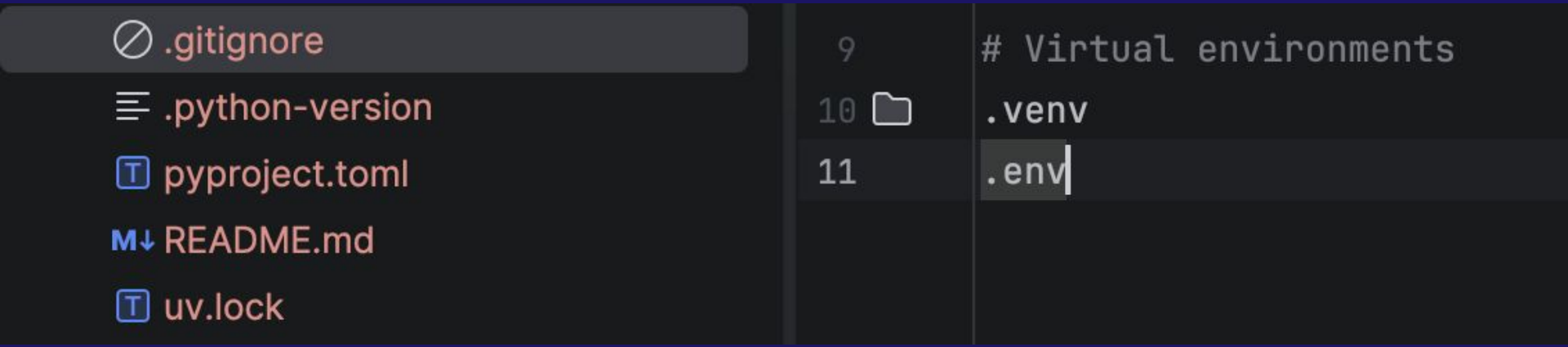
Environment Variables
(File Creation)



Environment Variables
(Results)



.gitignore
(example)



.env
(Content)


main.py
.gitignore
.env
1 DB_PORT="5432"
2 DB_PASSWORD="Benny_Banana_4"
3 DB_USERNAME="postgres"
4 DB_HOST="aws-1-eu-west-1.pooler.supabase.com"
5 DB_NAME="pfjscancbcuhplqbvyos"

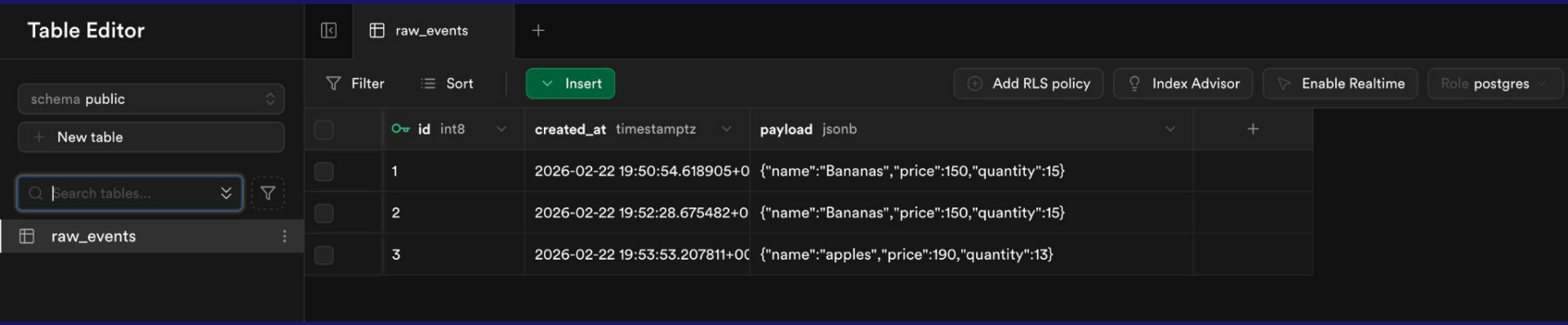
Environment Variables (Results)

```python
from dotenv import load_dotenv
import os

load_dotenv()  # reads .env from project root

DB_PORT = os.getenv("DB_PORT")
DB_USERNAME = os.getenv("DB_USERNAME")
DB_PASSWORD = os.getenv("DB_PASSWORD")
DB_NAME = os.getenv("DB_NAME")
DB_HOST = os.getenv("DB_HOST")
DATABASE_URL =
f"postgresql://{DB_USERNAME}.{DB_NAME}:{DB_PASSWORD}@{DB_HOST}:{DB_PORT}/postgres"
```

Function Example (Insert)

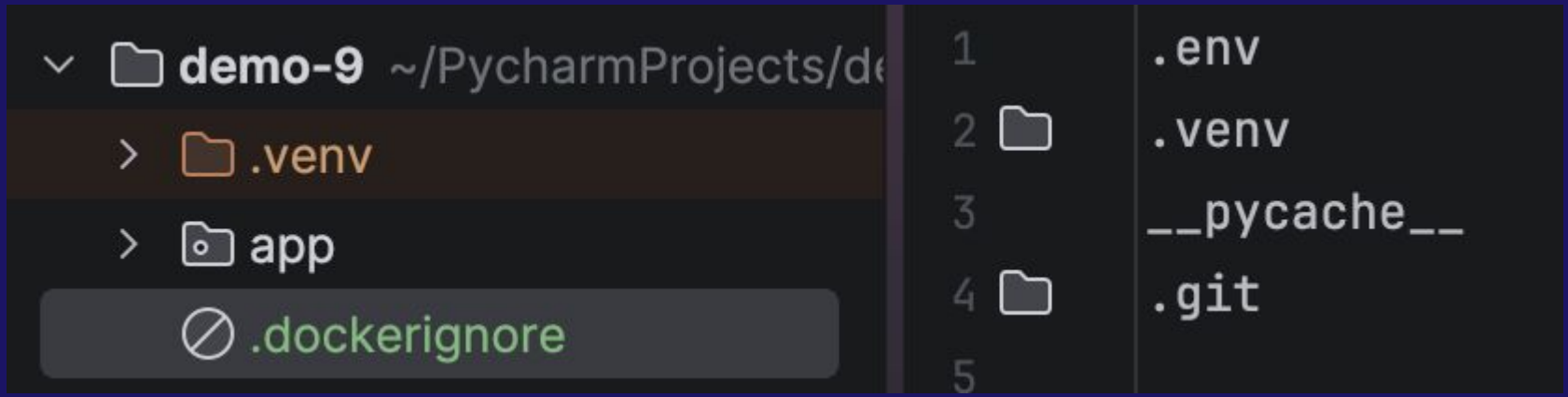
```python
@app.post("/product", status_code=status.HTTP_201_CREATED, response_model=Product)
def post_product(product: Product) -> Product:

    with pool.connection() as conn:
        with conn.transaction():
            conn.execute(
                "INSERT INTO raw_events (payload) VALUES (%s)",
                (json.dumps(product.model_dump()),)
            )

    return product
```

Results
(The End)




Docker
.env

.dockerIgnore
(gitignore but: for Docker)



Is docker installed?
(command)

```
$ docker run --env-file .env -p
8000:8000 fastapi-app
```

Injicera .env filen!

Is docker installed?
(command)

```
$ docker run --rm -it fastapi-app
sh -lc "ls -la /app | grep .env ||
echo 'No secrets exposed: OK'"
```

Verifiera att du inte råkat skriva in dina hemligheter

THANKS !

Do you have any questions?
kristoffer.johansson@sti.se