
GitHub Copilot for
ML/Data Ops

ML/Data Ops Use Cases

Schema Generation

Generate table **schemas** with relevant foreign key constraints


```
drop table if exists dbo.comments
drop table if exists dbo.likes
drop table if exists dbo.posts
drop table if exists dbo.users

create table dbo.users (
    id int identity(1,1) not null,
    name varchar(50) not null,
    primary key (id)
);

create table dbo.posts (
    id int identity(1,1) not null,
    title varchar(50) not null,
    content varchar(50) not null,
    user_id int not null,
    primary key (id),
    foreign key (user_id) references dbo.users(id)
);

create table dbo.comments (
    id int identity(1,1) not null,
    content varchar(50) not null,
    user_id int not null,
    post_id int not null,
    primary key (id),
    foreign key (user_id) references dbo.users(id),
    foreign key (post_id) references dbo.posts(id)
);

create table dbo.likes (
    id int identity(1,1) not null,
    user_id int not null,
    post_id int not null,
    primary key (id),
    foreign key (user_id) references dbo.users(id),
    foreign key (post_id) references dbo.posts(id)
);
```

Data Generation

Generate dummy data for development and testing


```sql
-- insert 100 random data into all existing tables in the database (users, posts, comments and likes) using loops
-- Language: sql
declare @i int = 1
declare @j int = 1
declare @k int = 1
declare @l int = 1

while @i <= 100
begin
    insert into dbo.users (name) values ('user' + cast(@i as varchar(50)))
    set @i = @i + 1
end

while @j <= 100
begin
    insert into dbo.posts (title, content, user_id) values ('title' + cast(@j as varchar(50)), 'content' + cast(@j as varchar(50)), @j)
    set @j = @j + 1
end

while @k <= 100
begin
    insert into dbo.comments (content, user_id, post_id) values ('comment' + cast(@k as varchar(50)), @k, @k)
    set @k = @k + 1
end

while @l <= 100
begin
    insert into dbo.likes (user_id, post_id) values (@l, @l)
    set @l = @l + 1
end;
```

SQL Query Generation

Generate SQL queries based on context and/or comments


```
select p.id, p.title, p.content, count(l.id) as likes_count
from dbo.posts p
left join dbo.likes l on p.id = l.post_id
```

```
-- show all posts order by likes count
-- Language: sql
select p.id, p.title, p.content, count(l.id) as likes_count
from dbo.posts p
left join dbo.likes l on p.id = l.post_id
group by p.id, p.title, p.content
order by likes_count desc

-- show all posts order by likes count using ctes
-- Language: sql
with cte as (
    select p.id, p.title, p.content, count(l.id) as likes_count
    from dbo.posts p
    left join dbo.likes l on p.id = l.post_id
    group by p.id, p.title, p.content
)
select * from cte order by likes_count desc
```

Library Imports

Auto import required libraries from initial input

```
# The model will use the Iris dataset to predict the species of an iris
# using a logistic regression classifier.

# Importing the necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
```

Comment Driven Development

Generate detailed code based on comments

```
# Let's visualize the distribution of the penguins species with a bar plot in matplotlib
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
fig = plt.figure(figsize=(10, 6))
ax = plt.axes()
ax.bar(df['Species'].unique(), df['Species'].value_counts())
ax.set_title('Number of penguins by species')
ax.set_xlabel('Species')
ax.set_ylabel('Number of penguins')
plt.show()
```

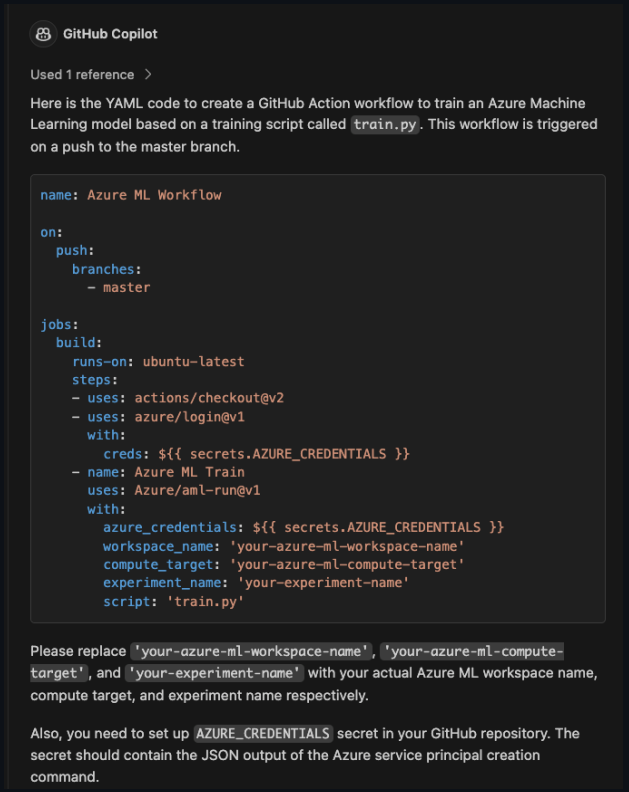
```
# Let's visualize with boxplots how the FlipperLength, CulmenLength and CulmenDepth are distributed for each species
# importing seaborn

import seaborn as sns
sns.set_style("whitegrid")
fig, ax = plt.subplots(1, 3, figsize=(20, 6))
sns.boxplot(x='Species', y='FlipperLength', data=df, ax=ax[0])
sns.boxplot(x='Species', y='CulmenLength', data=df, ax=ax[1])
sns.boxplot(x='Species', y='CulmenDepth', data=df, ax=ax[2])
plt.show()
```

✓ 0.4s

```

# Split the data into training and test sets in a way to have 30% of the data for testing

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.30, random_state=0)

```

DevOps

Use Copilot to generate your CI/CD deployment pipelines



Feature Engineering

Can generate features based on the columns and their relationships with each other

```
37 # feature engineering
36 # Create a new feature called CulmenRatio
35 df['CulmenRatio'] = df['CulmenLength'] / df['CulmenDepth']
34
33 # Create a new feature called FlipperRatio
32 df['FlipperRatio'] = df['FlipperLength'] / df['BodyMass']
31
30 # Create a new feature called BodyMassRatio
29 df['BodyMassRatio'] = df['BodyMass'] / df['CulmenDepth']
28
27 # Create a new feature called CulmenVolume
26 df['CulmenVolume'] = df['CulmenLength'] * df['CulmenDepth'] * df['CulmenDepth']
25
24 # Create a new feature called FlipperVolume
23 df['FlipperVolume'] = df['FlipperLength'] * df['FlipperLength'] * df['FlipperLength']
22
21 # Create a new feature called BodyMassVolume
20 df['BodyMassVolume'] = df['BodyMass'] * df['BodyMass'] * df['BodyMass']
19
18 # Create a new feature called BodyMassCulmenRatio
17 df['BodyMassCulmenRatio'] = df['BodyMass'] / df['CulmenLength']
16
```

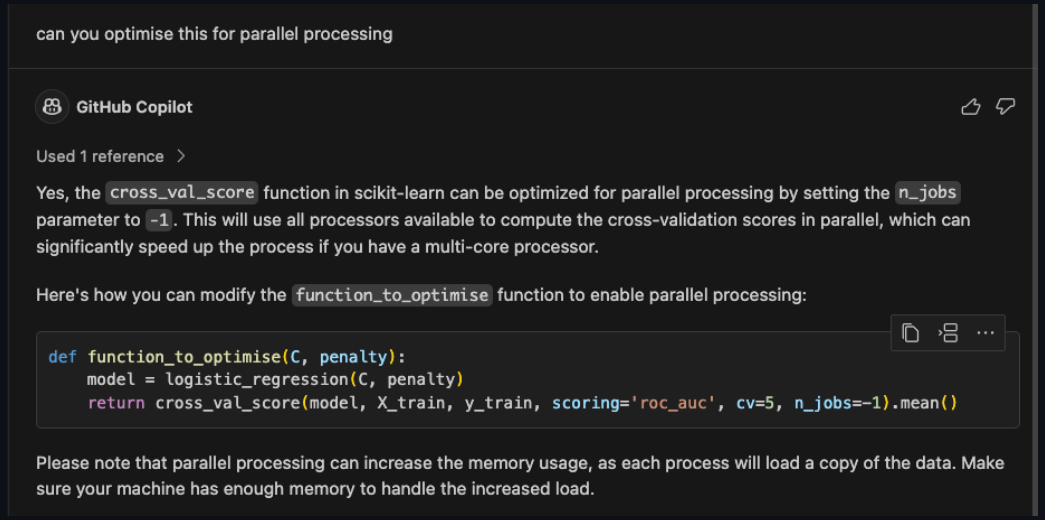
Hyperparameter tuning

Copilot can be used to help generate code for hyperparameter tuning

```python
# This file is used for hyperparameter turning using bayesian optimisation of a logistic regression model

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

from bayes_opt import BayesianOptimization
from bayes_opt import UtilityFunction

# Load data
X = pd.read_csv('X.csv')
y = pd.read_csv('y.csv')

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Define the model
def logistic_regression(C, penalty):
    model = Pipeline([
        ('scaler', StandardScaler()),
        ('classifier', LogisticRegression(C=C, penalty=penalty))
    ])
    return model

# Define the function to optimise

def function_to_optimise(C, penalty):
    model = logistic_regression(C, penalty)
    return cross_val_score(model, X_train, y_train, scoring='roc_auc', cv=5).mean()

# Define the bounds of the hyperparameters
pbounds = {'C': (0.01, 100), 'penalty': ('l1', 'l2')}
optimizer = BayesianOptimization(
    f=function_to_optimise,
    pbounds=pbounds,
    random_state=1,
)

# Optimise the model
optimizer.maximize(
    init_points=10,
    n_iter=100,
)

# Print the best hyperparameters
print(optimizer.max)
```

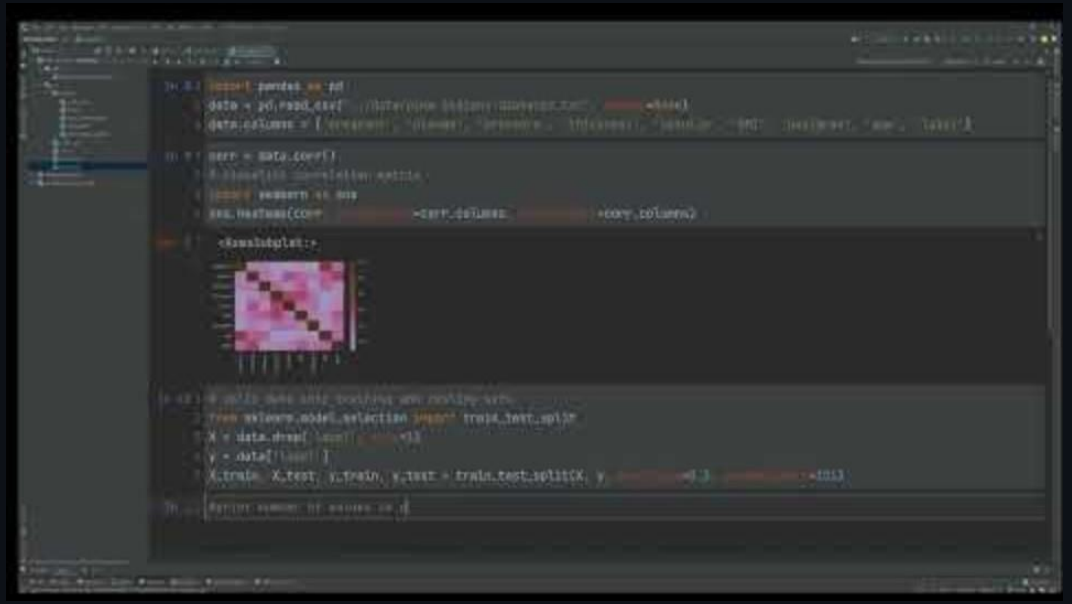
Parallel Processing Optimisation

Optimisation of code using parallel processing can be achieved



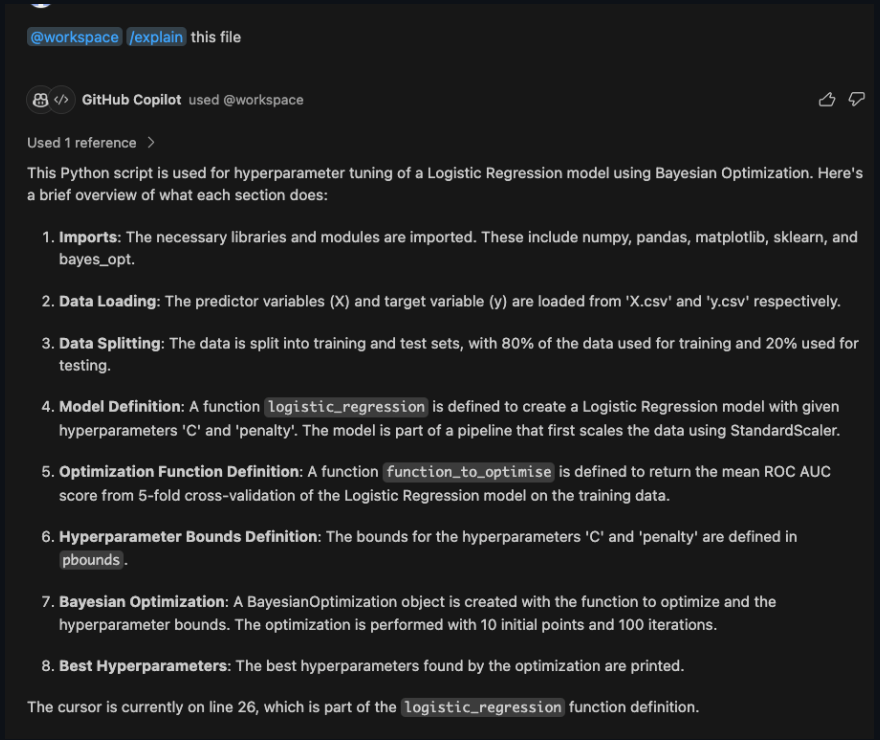
EDA (Exploratory Data Analysis)

Analyse a dataset and rapidly develop a basic model



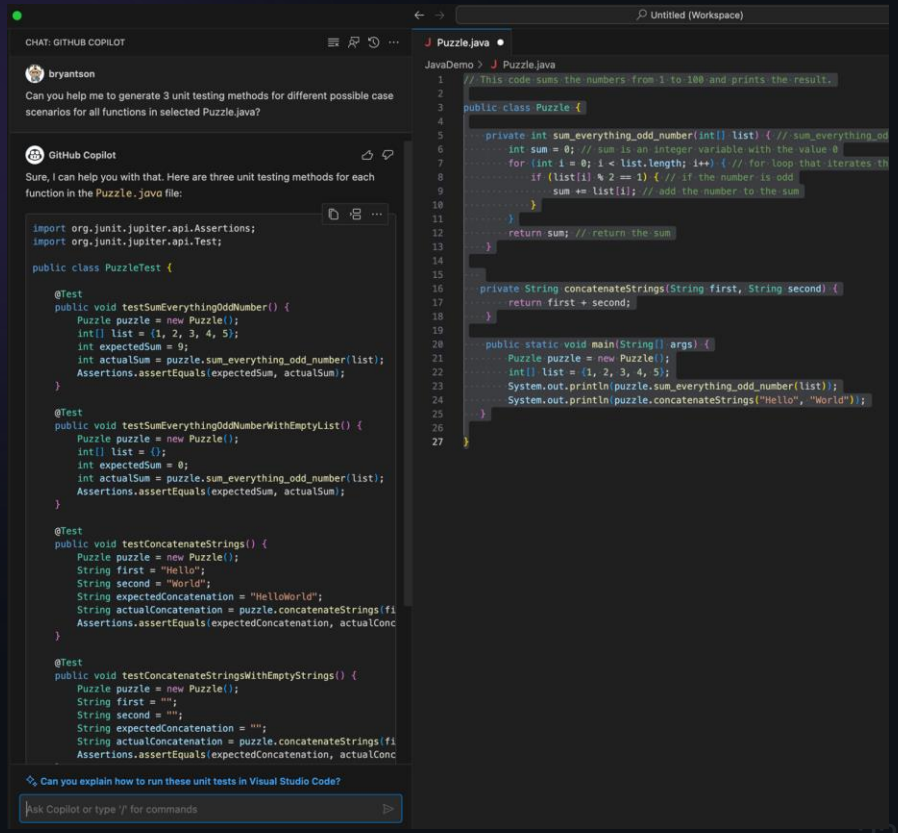
Understand complex code

Use of /explain can assist in making code easier to understand



Unit Testing Generation - Copilot Chat

Works best for general helper functions with well defined input and output



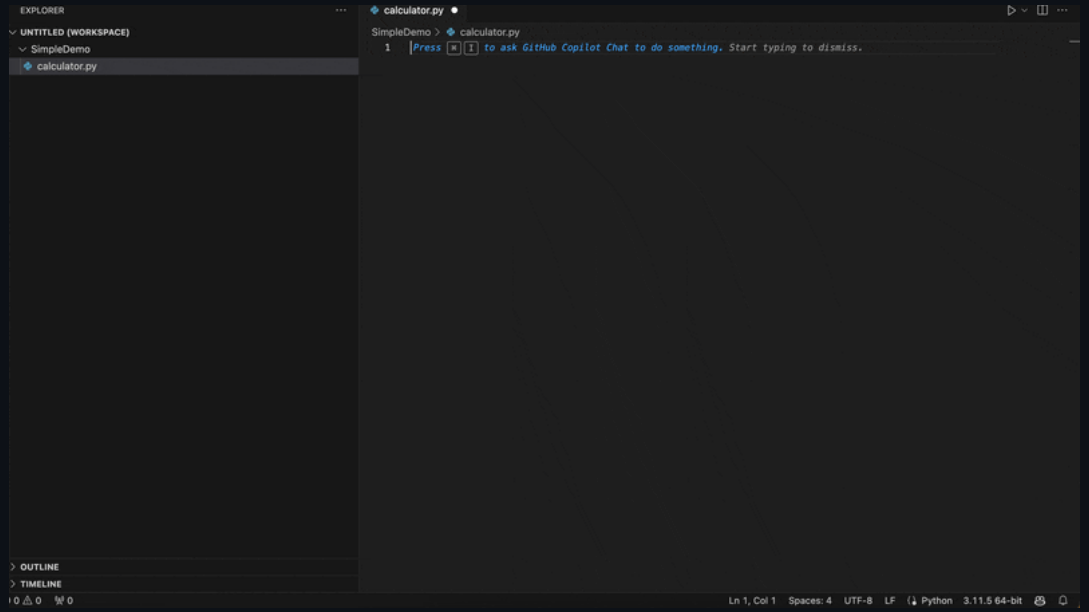
Common Problems

Problems #1:

Copilot fails to produce answer or keep repeating

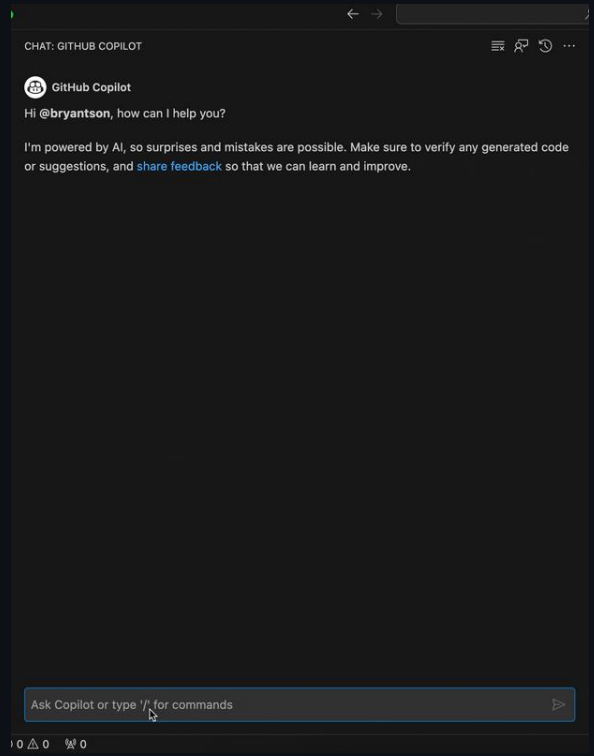
Some problems

- Fails to produce answer
- Hallucination - Keeps repeating



Problems #2:

Copilot generates incorrect result



Reasons why...

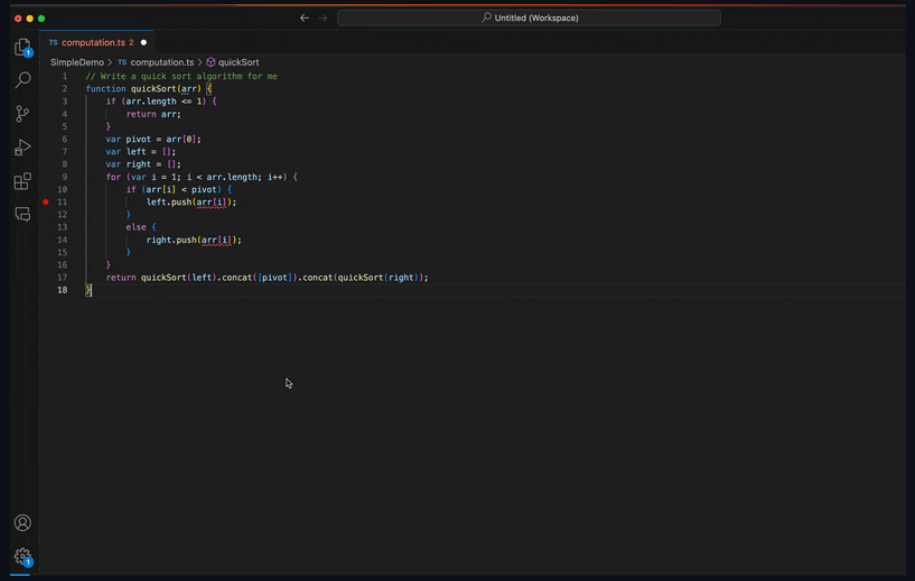
- Not enough context
- Old trained data
- Generative in nature

Problem #3:

Copilot suggests non-optimal solution

How to improve

- Ask in Copilot Chat
- Use in-line suggestion from Copilot



Lab

Frequently Asked Questions

“Is the information shared with GitHub Copilot secure?

“For example, let’s say I enter some sensitive information through GitHub Copilot, is it going to store the data like my user information?”

“Can I train GitHub Copilot on my private codebase?”

“Is the Copilot suggestion from an IDE based on my computer, a project opened in my IDE, files opened through tabs or just a single file?”

“What telemetry data can I get from GitHub Copilot ?”

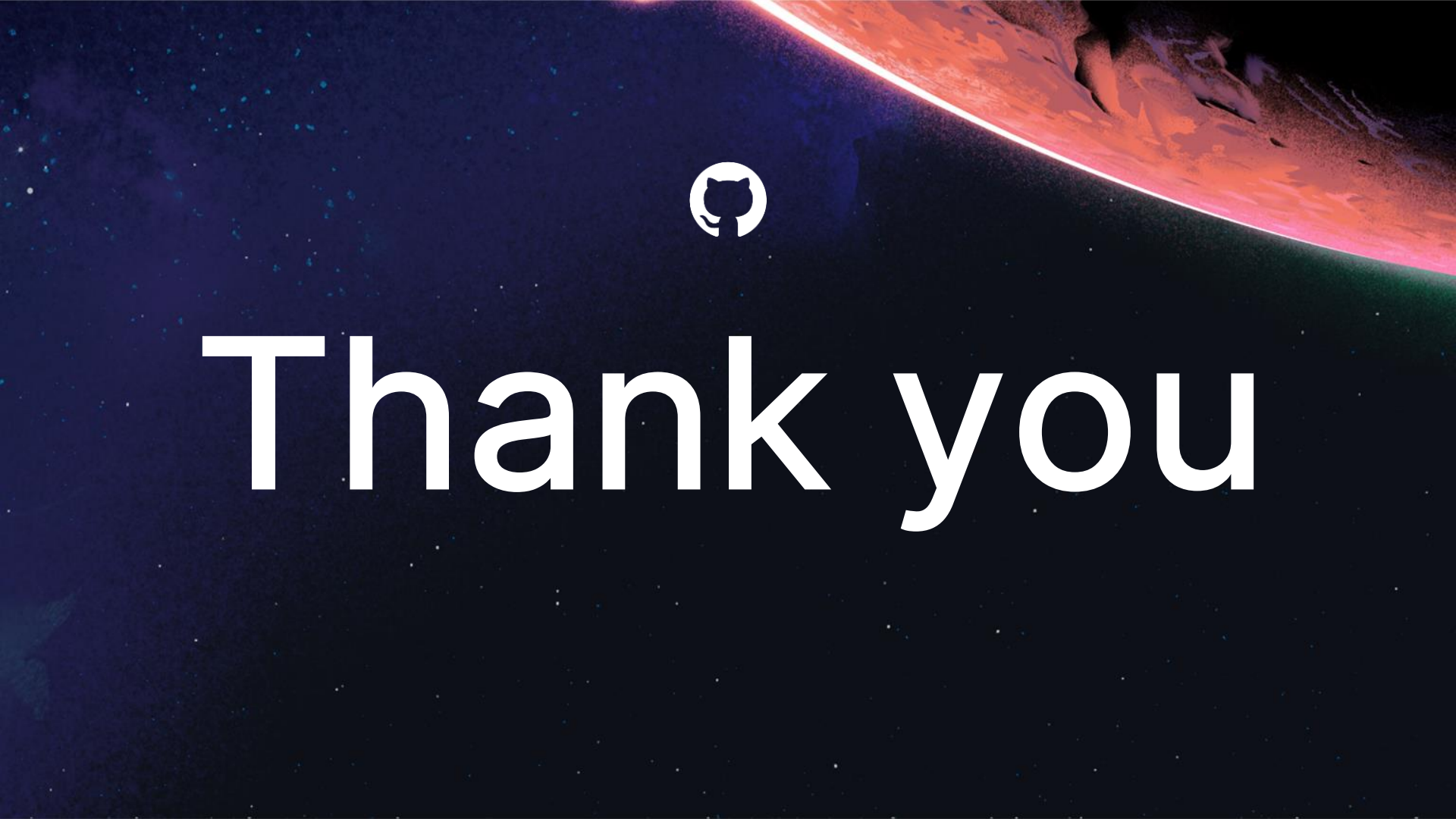
“How can GitHub Copilot integrate with existing pipelines/automations/scripts that we have?

“For example, I have this GitHub Actions/Azure DevOps pipeline, can GitHub Copilot integrate with those products so our code will always be secure?”

“Will Copilot replace my job”

“How much of my work can it take over?”

Q & A


Thank you