django devops Wozniak wanted

I have 8 technical questions for you, please fill this form :)

This helps us find the candidate we are looking for and save time on both sides !

Estimated time to complete is 20 minutes.

If you don't know the answer, please explain the reasoning on how you approach the specific problem.

If you pass the test you will be contacted on upwork.

Passion about engineering ? Love the nerdy pleasure of coding ?

Join a no BS tech company that won't lock you into boring daily meetings.

We are a friendly team of people who talk straight and love to work fast and hard.

* Indicates required question

What is your first and last name on Upwork ? (so we can match you) *

Your answer

what is your **email** address (in case we cannot find you on upwork. make sure there are no typos.) *

Your answer

1) Say you have 10 million rows in a db table represented by a django model. How do you efficiently load all of these, run a function on them and then write some changes back to the db. *

```
# Get the queryset for the model
queryset = MyModel.objects.all()

# Chunk the queryset into smaller chunks
chunks = queryset.chunk(100000)

# Iterate over the chunks and process the rows
for chunk in chunks:
    # Do something with the rows

# Write the changes back to the database
queryset.update()
```

2) When you run a database query, how do you make the database send result data to the client directly without loading all the result set onto onto its memory before sending it to client ? (client here being a script, not the browser) *

```
import mysql.connector

# Create a connection to the database
connection = mysql.connector.connect(
  host="localhost",
  user="root",
  password="password",
  database="my_database"
)

# Create a cursor
cursor = connection.cursor()

# Execute the query
cursor.execute("SELECT * FROM my_table")

# Get the result set
result_set = cursor.fetchall()

# Close the connection
connection.close()

# Iterate over the result set and print the rows
for row in result_set:
  print(row)
```

3) In production; service processes usually go down for many reasons. how to ensure they get restarted when that happens ? *

Here are some specific steps you can take to ensure that service processes get restarted when they go down:

Identify the critical services that need to be restarted. Not all services are created equal. Some services are more critical than others. Identify the services that are most critical to your business and make sure that they are the ones that are automatically restarted.
Configure your monitoring system to track the status of the critical services. Your monitoring system should be configured to send alerts when a critical service goes down. This will allow you to quickly identify the problem and take action to restart the service.
Configure your service orchestration tool to automatically restart the critical services. Your service orchestration tool should be configured to automatically restart critical services when they go down. This will ensure that the services are restarted quickly and without manual intervention.
Develop a process for manually restarting service processes. Even if you have a monitoring system and a service orchestration tool, there may be times when you need to manually restart a service process. Develop a process for manually restarting service

4) UserModel has many entries on it. the 3 UserModel queries run very slow. make needed changes and optimizations to UserModel to make the 3 queries run faster *

```
class UserModel(models.Model):

username = CharField(max_length=255)

role = JSONField() # format of ['admin', 'operator']

UserModel.objects.filter(username='john').first()

UserModel.objects.filter(username__contains='doe').first()

UserModel.objects.filter(role__contains='operator').first()
```

Here are the changes you can make to UserModel:

Add indexes on the username and role fields:

```
class UserModel(models.Model):
username = CharField(max_length=255)
role = JSONField() # format of ['admin', 'operator']

# Add indexes on the username and role fields
username_index = models.Index(fields=['username'])
role_index = models.Index(fields=['role'])
```

Use a caching library:

```
import django_cache

# Use a caching library
cache = django_cache.cache
```

Use a database that is optimized for performance:

```
# Use a database that is optimized for performance
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'my_database',
        'USER': 'my_user',
        'PASSWORD': 'my_password',
        'HOST': 'localhost',
        'PORT': 5432,
    }
}
```

5) we have 100 separate python scripts connecting at the same time to a central postgres database to perform multiple operations like reads and writes postgres creates a separate process for each of these clients, which is not efficient and consumes too much resources. how to make the central postgres db handle these concurrent connexions more efficiently ? *

Here are the steps on how to use a connection pool:

Choose a connection pool library. There are many different libraries available, and the best library for you will depend on the programming language that you are using.
Configure the library to use a connection pool. The configuration process will vary depending on the library that you are using.
Start using the library to connect to the database. When you connect to the database, the library will use a connection from the pool. When you disconnect from the database, the library will return the connection to the pool.

6) python is known to have a global interpreter lock, can you still run python code in parallel across multiple physical cores ? elaborate on the answer *

Yes, you can still run Python code in parallel across multiple physical cores, even though Python has a global interpreter lock (GIL). However, there are some limitations.

Here are some ways to run Python code in parallel across multiple physical cores:

Use the multiprocessing module.
Use a thread pool.
Use a Cython compiler.
The multiprocessing module allows you to create separate processes that can run in parallel. Each process has its own GIL, so multiple processes can execute Python bytecodes at the same time.

A thread pool is a group of threads that are waiting to be assigned work. When you submit a task to a thread pool, one of the threads in the pool will execute the task. This allows you to run multiple tasks in parallel, even though each task is running in a single thread.

Cython is a compiler that can compile Python code to C code. Cython can remove the GIL from your Python code, which can significantly improve the performance of your code.

However, it is important to note that the GIL is still present in Python, so there will be some overhead when running Python code in parallel. Additionally, not all Python code is able to be parallelized, so you will need to carefully consider your specific use case

7) our sysadmin team deploy our project from github onto two servers, usually this operation results in a 1 hour downtime. our estimates say we will need about 20 servers to handle the load during next months. we cannot afford to have 20 * 1 hours downtime to make new deployments, how can we efficiently deploy our projects onto these 20 servers with minimal downtime ? *

You can efficiently deploy your projects onto 20 servers with minimal downtime by using a blue/green deployment strategy. A blue/green deployment is a deployment strategy where you have two identical environments, one that is live and one that is staging. When you deploy a new version of your application, you deploy it to the staging environment first. Once you have verified that the new version is working correctly, you can then switch traffic from the live environment to the staging environment. This can help to minimize downtime because you only need to take one environment offline at a time.

8) (frontend question) In a large project with dozens of components, what would you do to ensure scalability?What things do you keep in mind while building a component? Finally, Use a third-party or go on with your own solution? How would you decide? *

Here are some things you can do to ensure scalability in a large project with dozens of components:

Use a component-based architecture. A component-based architecture is a way of designing your application where each component is responsible for a single task. This makes it easier to reuse components and to test your application.
Use a state management library. A state management library can help you to keep track of the state of your application. This can help to improve performance and to make your application more predictable.
Use a caching library. A caching library can help you to store frequently accessed data in memory. This can improve performance by reducing the number of times you need to access the database.
Use a load balancer. A load balancer can help you to distribute traffic across multiple servers. This can improve performance by ensuring that no single server is overloaded.
Here are some things to keep in mind while building a component:

Make sure the component is reusable. You should be able to use the component in multiple places in your application without having to make any changes.
Make sure the component is testable. You should be able to test the component in isolation without having to test the rest of your application.
Make sure the component is well-documented. You should provide documentation that explains how to use the component and how to test it.
Here are some factors to consider when deciding whether to use a third-party component or to build your own:

The cost of the third-party component. Third-party components can be expensive, especially if you need to license them for multiple developers.
The quality of the third-party component. Not all third-party components are created equal. Some components are poorly designed or have bugs.
The level of support for the third-party component. Some third-party component vendors provide excellent support, while others provide little or no support.
The level of customization you need. If you need to customize the component, you may be better off building your own.
Ultimately, the decision of whether to use a third-party component or to build your own is a trade-off between cost, quality, support, and customization.

Submit Clear form

Never submit passwords through Google Forms.