SparkPerf Benchmarking Platform User Manual

System Requirements:

- Docker and Docker Compose installed on your machine.

Installation and Setup:

1. Download and extract the Spark Benchmarking Tool package.
2. Create a new directory for your project and place the extracted package inside it.
3. Create a `requirements.txt` file in the same directory and add the following content:

```
Pyspark==3.0.0
numpy==1.22.0
Flask==2.1.1
findspark==2.0.1
Prometheus-client==0.8.0
```

4. Copy the necessary files: Download the following files to your local machine:

   - `docker-compose.yml`
   - `app.py`
   - `requirements.txt`

Configuration:

1. Open the `docker-compose.yml` file in a text editor.
2. Review and modify the configuration parameters based on your requirements.
    - Network Configuration: Modify the network settings if necessary.
    - Volume Configuration: Adjust the volume names or paths if needed.
    - Service Configuration: Customize the services according to your needs.

Starting the SparkPerf Benchmarking Tool:

Open a terminal or command prompt.

Navigate to the directory where you placed the Spark Benchmarking Tool package and the `docker-compose.yml` file.

Create a virtual environment by running the following command:

```
python3 -m venv env
```

This command creates a new virtual environment named "env" (you can choose a different name if desired).

Activate the virtual environment. The command varies depending on your operating system:

- For Windows:

```
env\Scripts\activate
```

- For macOS and Linux:

```
source env/bin/activate
```

Once the virtual environment is activated, install the required dependencies by running the following command:

```
pip install -r requirements.txt
```

This command installs the necessary packages specified in the `requirements.txt` file.

6. Start the Spark Benchmarking Tool by running the following command:

   ```
   docker-compose up -d
   ```

   This command starts the tool's services and creates the associated containers based on the provided configuration. Wait for the process to complete.

Accessing the Services:

- JupyterLab: Access JupyterLab by opening a web browser and navigating to `http://localhost:8888`.
- Spark Master: Access the Spark Master web UI by opening a web browser and going to `http://localhost:8080`.
- Spark Workers: Access the Spark Worker web UIs by opening a web browser and visiting `http://localhost:8081` and `http://localhost:8082`.
- Prometheus: Access Prometheus by opening a web browser and going to `http://localhost:9090`.
- Node Exporter: Access Node Exporter metrics by opening a web browser and visiting `http://localhost:9100`.
- cAdvisor: Access cAdvisor metrics by opening a web browser and navigating to `http://localhost:8085`.

- Grafana: Access Grafana by opening a web browser and visiting `http://localhost:3000`. Log in using the admin credentials specified in the environment variables.

Stopping the Spark Benchmarking Tool:

1. Open a terminal or command prompt.
2. Navigate to the directory containing the `docker-compose.yml` file.
3. Run the following command to stop and remove the containers:
4. Copy code
5.

   This command stops and removes the containers associated with the services.

Make sure to place this `app.py` file in the same directory as your Spark Benchmarking Tool package.

This `app.py` file includes the following:

- Logging configuration setup.
- Initialization of Spark and the necessary imports.
- Definition of the `get_spark_session()` function to create and configure a Spark session.
- Flask application setup.
- A route `/get/` that performs a Spark test using the provided CSV file and returns a sample result.
- Exception handling to capture and display any exceptions thrown during the Spark test.
- Flask application running on `0.0.0.0` (all network interfaces) and port `5001`.

After you make sure, all the component are up and running, you need to run the workload script.

Here's an explanation of the setup of Django admin panel on a VM for your user manual:

1. Accessing the VM:
    - Connect to the VM using your preferred method, such as SSH or remote desktop.
2. Installing Django:
    - Check if Django is already installed on the VM by running the command: `python3 -m django --version`.
    - If Django is not installed, you can install it using the command: `pip3 install django`.

3. Setting up the Django project:
    - Create a new Django project using the command: `django-admin startproject benchmarking_panel`.
    - Navigate into the project directory: `cd benchmarking_panel`.

4. Creating a Django app:
    - Create a new Django app within the project using the command: `python3 manage.py startapp benchmarking_app`.

5. Defining the model:
    - Open the file `benchmarking_app/models.py` and define the model that represents your configuration information.
    - Customize the fields in the model to match your configuration requirements. For example, you can define fields like `name`, `value`, `description`, etc.
    - Optionally, you can add any necessary methods or additional model customization as per your needs.

6. Registering the model in the admin panel:
    - Open the file `benchmarking_app/admin.py`.
    - Import your model by adding the line: `from .models import YourModel`.
    - Register your model in the admin panel by adding the following line:

7. Applying migrations:

- Apply the necessary database migrations to create the table for your model using the command: `python3 manage.py makemigrations` followed by `python3 manage.py migrate`.

8. Creating a superuser:
    - Create a superuser account to access the admin panel using the command: `python3 manage.py createsuperuser`.
    - Follow the prompts to set a username, email, and password for the superuser.
9. Starting the Django development server:
    - Start the Django development server using the command: `python3 manage.py runserver 0.0.0.0:8000`.
10. Accessing the Django admin panel:
    - Open a web browser on your local machine.
    - Enter the IP address or hostname of the VM followed by `:8000/admin` in the address bar. For example: `http://<VM_IP_address>:8000/admin`.
    - Log in with the superuser credentials created earlier.
11. Using the Django admin panel:
    - Once logged in, you will see the Django admin panel interface.
    - Locate your model in the admin panel (it should be listed under the registered models).
    - Click on your model to view the configuration information.
    - Use the provided interface to add, edit, or delete configuration entries.
    - You can search, filter, and sort the configuration entries based on the defined fields.
    - Customize the admin panel further by modifying the model's admin options, such as list display, search fields, fieldsets, etc. Refer to the Django documentation for more details on customizing the admin panel.
12. Updating and managing configuration information:
    - Use the admin panel interface to update and manage the configuration information stored in your model.
    - Add new configuration entries by clicking the "Add" button and filling in the required fields.
    - Edit existing entries by clicking on the entry and updating the field values.
    - Delete entries by selecting the entries and clicking the "Delete" button.

To generate data, start by accessing the "Datas" section on the Django server. Look for the "Add data" option located at the top right-hand corner. In this section, you can input the desired configuration of your data, including the "Desired size," the current size, and upload your sample data. Once you have entered the necessary details, click on the "SAVE" button.

Now, return to the main page where you will find the status of your data. To proceed, select your data by clicking on the small square next to it. Then, navigate to the "Action" tab and choose the "convert data" option. Finally, click on the "go" button to initiate the data generation process.