SPEED Deployment System API Documentation

The SPEED Deployment System API is designed to manage and track the deployment processes of software projects, leveraging Docker containers for execution. This document provides a comprehensive overview of the API endpoints, including their functionality, request methods, input parameters, and expected responses.

System Initialization

Pre-requisites

- A SQLite database named deployments.db must be accessible.
- Docker must be installed and running on the host machine.
- The utils.py module, containing the DBManager class and run_docker_container function, must be correctly set up.

Initialization

Upon the first request, the system will automatically initialize by creating necessary tables in the database to track deployments and their results.

Deployment Status

The status of a deployment can vary based on its current state, such as Pending, Running, Succeeded, or Failed. The exact status values are managed internally and may be updated based on the system's operational logic.

Running the Application

To run the SPEED Deployment System API, ensure that all prerequisites are met, then execute the Python script hosting the Flask application:
python main.py

Ensure the Docker daemon is running, and the database file (deployments.db) is correctly placed and accessible.

API Endpoints

POST /start

Description

Initiates a new deployment process by building and executing the specified repository in a Docker container. This endpoint generates a unique worker ID for the deployment.

Request Body

- url (string): The URL of the Git repository to deploy.
- branch (string): The branch of the Git repository to deploy.

Response

- Status Code: 201
- Body:

```
{
 "id": "<deployment_id>"
}
```

- id (int): The unique ID of the newly created deployment.

GET /info/int:deployment_id

Description

Retrieves information about a specific deployment, including the repository name, branch, and current status.

URL Parameters

- deployment_id (int): The unique ID of the deployment.

Response

- Status Code: 200 (OK) or 404 (Not Found)
- Body (if found):

```
{
 "id": "<deployment_id>",
 "repo_name": "<repository_url>",
 "repo_branch": "<branch>",
 "status": "<status>"
}
```

- Body (if not found):

```
{
 "error": "Deployment not found"
}
```

POST /update

Description

Updates the status of an existing deployment. This endpoint is typically used by the deployment leaders to report the progress or completion of a deployment.

Request Body

- id (int): The ID of the deployment to update.
- new_results (string): The new results or status to update the deployment with.

Response

- Status Code: 200
- Body:


```
{
 "message": "Deployment updated successfully"
}
```

POST /add_results/int:deployment_id

Description

Adds results for a specific deployment. This can include logs, output data, or any relevant information regarding the deployment's outcome.

URL Parameters

- deployment_id (int): The unique ID of the deployment.

Request Body

- results (string): The results or data to add for the deployment.

Response

- Status Code: 200 (OK) or 400 (Bad Request)
- Body (if successful):

```
{
 "message": "Results added successfully"
}
```

- Body (if failure):

```
{
 "error": "Missing results data"
}
```

GET /results/int:deployment_id

Description

Retrieves all results associated with a specific deployment.

URL Parameters

- deployment_id (int): The unique ID of the deployment.

Response

- Status Code: 200
- Body:

```
{
 "results": "<results_data>"
}
```