Knowledge Transfer (KT) Document for Python REST API with Multiple Endpoints

Overview

This document provides a detailed explanation of each endpoint in the Python codebase, its purpose, functionality, and technical implementation. The system is designed to interact with Elasticsearch, MongoDB, and other services (Ansible, LB,)for event data processing, reporting, and pipeline orchestration.

Endpoints

1. /init-build-report

- Purpose: Fetches VM-related build reports within a specific time window (last 15 minutes) and submits the data to an Elasticsearch pipeline.

- Functionality:

1. Calculates start\_time and end\_time based on the current UTC time.

2. Constructs a URL to query VM events from an external service.

3. Extracts relevant event IDs from the response using the \_ingest() method.

4. Submits the data to the PollVMEvents pipeline.

- Output: Returns a message confirming the pipeline initiation.

2. /datadump-pipeline

- Purpose: Dumps data from Elasticsearch for a specified time range into MongoDB.

- Parameters:

- elk\_index: Name of the Elasticsearch index to query.

- duration: Time range in hours for the query.

- end\_date (optional): Specifies the query's end time.

- Functionality:

1. Validates the input parameters using reqparse.

2. Computes start\_time and end\_time based on the provided duration.

3. Fetches data from Elasticsearch using the specified index and time range.

4. Submits the data to the DumpElkBuilds pipeline for storage in MongoDB.

- Output: Success message indicating pipeline initiation.

3. /datadump-pipeline\_via\_id

- Purpose: Similar to /datadump-pipeline, but retrieves specific data for a given resource ID.

- Parameters:

- elk\_index: Elasticsearch index to query.

- id: Specific resource ID for data retrieval.

- Functionality:

1. Validates inputs.

2. Queries Elasticsearch for data matching the resource ID.

3. Pushes the fetched data into MongoDB.

- Output: Confirms successful pipeline execution.

4. /run-ansible-workflow

- Purpose: Triggers an Ansible job across regions such as AMRS, APAC, and EMEA.

- Parameters:

- env: Specifies the environment (e.g., PROD, UAT).

- Functionality:

1. Configures API URLs and tokens for each region based on the env parameter.

2. Submits the configuration to the PollAnsible pipeline.

- Output: Success message confirming job initiation.

5. /run-tower-report

- Purpose: Fetches and processes tower-related reports from various regions.

- Functionality:

1. Prepares a list of tower URLs and tokens.

2. Queries data from each tower and processes the responses.

- Output: Confirms report generation.

6. /run-lb-info-workflow

- Purpose: Processes load balancer (LB) event data.

- Functionality:

1. Queries Elasticsearch for LB events.

2. Submits the data to the PollLoadBalancer pipeline.

- Output: Success message with the number of processed events.

7. /run-lb-create-workflow and /run-lb-update-workflow

- Purpose:

- Create Workflow: Handles load balancer creation events.

- Update Workflow: Handles load balancer update events.

- Functionality:

1. Queries Elasticsearch for the respective event type.

2. Submits the data to the appropriate pipeline (PollLoadBalancerCreate or PollLoadBalancerUpdate).

- Output: Confirmation of successful pipeline initiation.

8. /run-heatmap-pipeline

- Purpose: Generates heatmaps for VM utilization using event data.

- Functionality:

1. Fetches VM events from Elasticsearch.

2. Uses the HeatMapReducer pipeline to compute VM usage statistics.

3. Stores the processed data in MongoDB.

- Output: Success message indicating heatmap generation.

9. /run-bmp-day1-build-pipeline

- Purpose: Processes Day 1 Build Management Pipeline (BMP) events.

- Pipeline: PollBMPDay1Events.

- Output: Confirmation of pipeline initiation.

10. /run-bmp-day2-build-pipeline

- Purpose: Processes Day 2 BMP events.

- Pipeline: PollBMPDay2Events.

- Output: Confirmation of pipeline initiation.

11. /update-vm-build-via-resourceid

- Purpose: Updates VM build details for a specific resource ID.

- Parameters:

- res\_id: Resource ID to update.

- Functionality:

1. Fetches res\_id from the request.

2. Queries Elasticsearch for the resource.

3. Updates MongoDB with the fetched data.

- Output: Success message confirming the update.

12. /update-vm-day1\_build-via-resourceid

- Purpose: Updates Day 1 VM build data for a specific resource ID.

- Pipeline: PollVMDay1Events.

13. /update-vm-day2\_build\_mongo-via-resourceid

- Purpose: Updates Day 2 VM build data in MongoDB for a specific resource ID.

- Pipeline: PollVMDay2Events.

14. /vm-build-report

- Purpose: Fetches and processes VM build data for a specified environment.

- Parameters:

- env: Specifies the environment (e.g., PROD, UAT).

- Functionality:

1. Queries Elasticsearch for VM build events.

2. Submits the data to MongoDB for storage.

15. /vm-day1-build-report

- Purpose: Processes VM Day 1 build reports.

- Pipeline: PollVMDay1Events.

16. /vm-day2-build-report

- Purpose: Handles VM Day 2 build reports.

- Pipeline: PollVMDay2Events.

Common Functional Workflow

1. Input Parsing:

- Uses reqparse to validate inputs like elk\_index, res\_id, and env.

- Example:

python

parser\_env.add\_argument('env', required=True, type=str, choices=["PROD", "UAT"])

args = parser\_env.parse\_args()

Explanation of Parsers Used

- parser\_env: Validates and ensures the input env is one of the allowed values (PROD, UAT). Used for environment-specific endpoints like /vm-build-report.

- parser\_id: Validates and ensures a res\_id (resource ID) is provided. This is commonly used in update endpoints like /update-vm-build-via-resourceid.

- parser\_datadump: Ensures parameters like elk\_index, duration, and end\_date are correctly passed for fetching a time range of data from Elasticsearch. Used in endpoints like /datadump-pipeline.

- parser\_datadump\_via\_id: Used to validate inputs for fetching data by a specific resource ID (id) in /datadump-pipeline\_via\_id.

Detailed Class Breakdown

1. BuildReport Class

- Purpose: Handles the /init-build-report endpoint.

- Key Methods:

- get(self): Fetches VM-related events within the last 15 minutes and submits them to a pipeline.

- \_ingest(res): Extracts event IDs from the API response.

- Logic:

1. Fetches data using a pre-configured URL.

2. Processes the JSON response to extract relevant fields.

3. Submits the extracted data to the PollVMEvents pipeline.

- Output: Confirms pipeline initiation.

2. DataDumpPipeline Class

- Purpose: Handles the /datadump-pipeline endpoint.

- Key Methods:

- get(self): Fetches logs from Elasticsearch for a given time range and submits them to MongoDB.

- \_pollElkData(): Helper method for querying Elasticsearch.

- Logic:

1. Validates parameters such as elk\_index and duration.

2. Queries Elasticsearch for the specified time range.

3. Dumps the fetched data into MongoDB using the DumpElkBuilds pipeline.

- Output: Confirms pipeline initiation.

3. DataDumpPipelineViaID Class

- Purpose: Handles the /datadump-pipeline\_via\_id endpoint.

- Key Methods:

- get(self): Fetches Elasticsearch logs by resource ID and stores them in MongoDB.

- Logic:

1. Validates input parameters (elk\_index, id).

2. Queries Elasticsearch for the specific resource ID.

3. Submits the fetched data to the MongoDB pipeline.

4. AnsibleDashboard Class

- Purpose: Handles the /run-ansible-workflow endpoint.

- Key Methods:

- get(self): Initiates an Ansible job based on the specified region and environment.

- Logic:

1. Configures API tokens and URLs for different regions (AMRS, APAC, EMEA).

2. Submits the configuration to the PollAnsible pipeline.

- Output: Confirms the Ansible job initiation.

5. TowerReport Class

- Purpose: Handles the /run-tower-report endpoint.

- Key Methods:

- get(self): Fetches and processes tower reports for specified regions.

- Logic:

1. Uses pre-configured URLs and tokens to query tower data.

2. Processes the data for reporting.

6. RolesPipeline Class

- Purpose: Handles the /run-roles-pipeline endpoint.

- Key Methods:

- get(self): Maps roles to accounts using MongoDB.

- Logic:

1. Initiates a pipeline to map roles to accounts.

2. Stores the results in MongoDB.

- Output: Confirms the job initiation.

7. LBInfoReport Class

- Purpose: Handles the /run-lb-info-workflow endpoint.

- Key Methods:

- get(self): Fetches load balancer event data from Elasticsearch.

- Logic:

1. Queries Elasticsearch for load balancer events.

2. Processes the data and submits it to the PollLoadBalancer pipeline.

8. LBCreateReport and LBUpdateReport Classes

- Purpose: Handles creation and update workflows for load balancers.

- Key Methods:

- get(self): Fetches and processes load balancer creation or update events.

- Logic:

1. Queries Elasticsearch for the respective event type.

2. Submits the data to the relevant pipeline (PollLoadBalancerCreate or PollLoadBalancerUpdate).

9. VMHeatMapPipeline Class

- Purpose: Handles the /run-heatmap-pipeline endpoint.

- Key Methods:

- get(self): Processes VM event data to generate a utilization heatmap.

- Logic:

1. Fetches VM events from Elasticsearch.

2. Uses the HeatMapReducer pipeline to compute VM usage statistics.

3. Stores the processed data in MongoDB.

10. VMBuildReport Class

- Purpose: Handles the /vm-build-report endpoint.

- Key Methods:

- get(self): Fetches and processes VM build reports for a specific environment.

- Logic:

1. Validates the env parameter.

2. Queries Elasticsearch for VM build events.

3. Submits the processed data to MongoDB.

11. VMDay1BuildReport and VMDay2BuildReport Classes

- Purpose: Handles Day 1 and Day 2 VM build reports.

- Key Methods:

- get(self): Processes Day 1 or Day 2 VM events.

- Logic:

1. Queries Elasticsearch for Day 1 or Day 2 VM events.

2. Submits the processed data to MongoDB using PollVMDay1Events or PollVMDay2Events.

12. UpdateVMBuildViaResourceID Class

- Purpose: Handles the /update-vm-build-via-resourceid endpoint.

- Key Methods:

- get(self): Updates VM build details for a specific resource ID.

- Logic:

1. Fetches res\_id from the request.

2. Queries Elasticsearch for the VM data.

3. Updates the MongoDB record.

13. UpdateVMDay1BuildViaResourceID and UpdateVMDay2BuildViaResourceID Classes

- Purpose: Handles updates for Day 1 and Day 2 VM build data using resource IDs.

- Key Methods:

- get(self): Updates MongoDB with the respective VM event data.

- Logic:

1. Validates the res\_id.

2. Queries Elasticsearch for the Day 1 or Day 2 VM data.

3. Submits the data to the relevant pipeline.

Conclusion

This document explains the detailed functionality of all endpoints, their inputs, processing logic, and outputs. The codebase is a robust system for managing pipelines across various event types, offering flexibility and scalability.