i have awsSdk\_ApiCall\_ApiCallDuration\_seconds, awsSdk\_ApiCall\_ApiCallDuration\_seconds\_buckets, awsSdk\_ApiCall\_ApiCallDuration\_seconds\_count, awsSdk\_ApiCall\_ApiCallDuration\_seconds\_max, awsSdk\_ApiCall\_CredentialsFetchDuration\_seconds, awsSdk\_ApiCall\_CredentialsFetchDuration\_seconds\_bucket, awsSdk\_ApiCall\_CredentialsFetchDuration\_seconds\_count, awsSdk\_ApiCall\_CredentialsFetchDuration\_seconds\_max, awsSdk\_ApiCall\_CredentialsFetchDuration\_seconds\_sum, awsSdk\_ApiCall\_EndpointResolveDuration\_seconds, awsSdk\_ApiCall\_EndpointResolveDuration\_seconds\_bucket, awsSdk\_ApiCall\_EndpointResolveDuration\_seconds\_count, awsSdk\_ApiCall\_EndpointResolveDuration\_seconds\_max, awsSdk\_ApiCall\_EndpointResolveDuration\_seconds\_sum, awsSdk\_ApiCall\_MarshallingDuration\_seconds\_bucket, awsSdk\_ApiCall\_MarshallingDuration\_seconds, awsSdk\_ApiCall\_MarshallingDuration\_seconds\_count, awsSdk\_ApiCall\_MarshallingDuration\_seconds\_max, awsSdk\_ApiCall\_MarshallingDuration\_seconds\_sum, awsSdk\_ApiCall\_RetryCount, awsSdk\_ApiCall\_RetryCount\_counter\_total, awsSdk\_ApiCallAttempt\_BackoffDelayDuration\_seconds, awsSdk\_ApiCallAttempt\_BackoffDelayDuration\_seconds\_bucket, awsSdk\_ApiCallAttempt\_BackoffDelayDuration\_seconds\_count, awsSdk\_ApiCallAttempt\_BackoffDelayDuration\_seconds\_max, awsSdk\_ApiCallAttempt\_BackoffDelayDuration\_seconds\_sum, awsSdk\_ApiCallAttempt\_HttpStatusCode, awsSdk\_ApiCallAttempt\_HttpStatusCode\_counter\_total, awsSdk\_ApiCallAttempt\_ServiceCallDuration\_seconds, awsSdk\_ApiCallAttempt\_ServiceCallDuration\_seconds\_bucket, awsSdk\_ApiCallAttempt\_ServiceCallDuration\_seconds\_count, awsSdk\_ApiCallAttempt\_ServiceCallDuration\_seconds\_sum, awsSdk\_ApiCallAttempt\_SigningDuration\_seconds, awsSdk\_ApiCallAttempt\_SigningDuration\_seconds\_bucket, awsSdk\_ApiCallAttempt\_SigningDuration\_seconds\_count, awsSdk\_ApiCallAttempt\_SigningDuration\_seconds\_max, awsSdk\_ApiCallAttempt\_SigningDuration\_seconds\_sum, awsSdk\_ApiCallAttempt\_TimeToFirstByte\_seconds, awsSdk\_ApiCallAttempt\_TimeToFirstByte\_seconds\_bucket, awsSdk\_ApiCallAttempt\_TimeToFirstByte\_seconds\_count, awsSdk\_ApiCallAttempt\_TimeToFirstByte\_seconds\_sum, awsSdk\_ApiCallAttempt\_TimeToFirstByte\_seconds\_sum, awsSdk\_ApiCallAttempt\_TimeToFirstByte\_seconds\_max, awsSdk\_ApiCallAttempt\_TimeToLastByte\_seconds\_bucket, awsSdk\_ApiCallAttempt\_TimeToLastByte\_seconds\_count, awsSdk\_ApiCallAttempt\_TimeToLastByte\_seconds\_max, awsSdk\_ApiCallAttempt\_TimeToLastByte\_seconds\_sum i want to create reasonable grafana singleview dashboard for the s3 buckets which uses these metrics, what are the best suitable dashboard to create and which of the metrics combinations will achieve that using grafana PromQL

**Key Panels and Metrics**

**1. API Call Duration**

* **Purpose**: Measure how long API calls take.
* **Metrics**:
  + awsSdk\_ApiCall\_ApiCallDuration\_seconds
  + awsSdk\_ApiCall\_ApiCallDuration\_seconds\_bucket
  + awsSdk\_ApiCall\_ApiCallDuration\_seconds\_max
* **Panel Type**: Line or Heatmap
* **PromQL**:
  + Average duration:

promql

Copy code

avg(rate(awsSdk\_ApiCall\_ApiCallDuration\_seconds\_sum[5m]) / rate(awsSdk\_ApiCall\_ApiCallDuration\_seconds\_count[5m]))

* + Maximum duration:

promql

Copy code

max\_over\_time(awsSdk\_ApiCall\_ApiCallDuration\_seconds\_max[5m])

**2. Credentials Fetch Duration**

* **Purpose**: Monitor time taken to fetch credentials, which affects API performance.
* **Metrics**:
  + awsSdk\_ApiCall\_CredentialsFetchDuration\_seconds
  + awsSdk\_ApiCall\_CredentialsFetchDuration\_seconds\_sum
  + awsSdk\_ApiCall\_CredentialsFetchDuration\_seconds\_max
* **Panel Type**: Line Chart
* **PromQL**:
  + Average credentials fetch time:

promql

Copy code

avg(rate(awsSdk\_ApiCall\_CredentialsFetchDuration\_seconds\_sum[5m]) / rate(awsSdk\_ApiCall\_CredentialsFetchDuration\_seconds\_count[5m]))

* + Maximum credentials fetch time:

promql

Copy code

max\_over\_time(awsSdk\_ApiCall\_CredentialsFetchDuration\_seconds\_max[5m])

**3. Endpoint Resolution Duration**

* **Purpose**: Visualize how long it takes to resolve endpoints for API calls.
* **Metrics**:
  + awsSdk\_ApiCall\_EndpointResolveDuration\_seconds
  + awsSdk\_ApiCall\_EndpointResolveDuration\_seconds\_sum
* **Panel Type**: Line Chart
* **PromQL**:
  + Average endpoint resolution time:

promql

Copy code

avg(rate(awsSdk\_ApiCall\_EndpointResolveDuration\_seconds\_sum[5m]) / rate(awsSdk\_ApiCall\_EndpointResolveDuration\_seconds\_count[5m]))

**4. Marshalling Duration**

* **Purpose**: Assess how long it takes to serialize requests.
* **Metrics**:
  + awsSdk\_ApiCall\_MarshallingDuration\_seconds
  + awsSdk\_ApiCall\_MarshallingDuration\_seconds\_sum
* **Panel Type**: Bar Chart
* **PromQL**:
  + Average marshalling time:

promql

Copy code

avg(rate(awsSdk\_ApiCall\_MarshallingDuration\_seconds\_sum[5m]) / rate(awsSdk\_ApiCall\_MarshallingDuration\_seconds\_count[5m]))

**5. Retry Count**

* **Purpose**: Identify the frequency of retries for API calls, indicating transient failures.
* **Metrics**:
  + awsSdk\_ApiCall\_RetryCount\_counter\_total
* **Panel Type**: Bar Chart
* **PromQL**:
  + Total retries per minute:

promql

Copy code

rate(awsSdk\_ApiCall\_RetryCount\_counter\_total[1m])

**6. Service Call Duration**

* **Purpose**: Monitor latency introduced by service calls to S3.
* **Metrics**:
  + awsSdk\_ApiCallAttempt\_ServiceCallDuration\_seconds
  + awsSdk\_ApiCallAttempt\_ServiceCallDuration\_seconds\_sum
  + awsSdk\_ApiCallAttempt\_ServiceCallDuration\_seconds\_count
* **Panel Type**: Line Chart
* **PromQL**:
  + Average service call duration:

promql

Copy code

avg(rate(awsSdk\_ApiCallAttempt\_ServiceCallDuration\_seconds\_sum[5m]) / rate(awsSdk\_ApiCallAttempt\_ServiceCallDuration\_seconds\_count[5m]))

**7. Time to First Byte**

* **Purpose**: Evaluate time taken to receive the first byte of a response.
* **Metrics**:
  + awsSdk\_ApiCallAttempt\_TimeToFirstByte\_seconds
  + awsSdk\_ApiCallAttempt\_TimeToFirstByte\_seconds\_sum
* **Panel Type**: Line Chart
* **PromQL**:
  + Average time to first byte:

promql

Copy code

avg(rate(awsSdk\_ApiCallAttempt\_TimeToFirstByte\_seconds\_sum[5m]) / rate(awsSdk\_ApiCallAttempt\_TimeToFirstByte\_seconds\_count[5m]))

**8. HTTP Status Codes**

* **Purpose**: Analyze API call success and failure rates.
* **Metrics**:
  + awsSdk\_ApiCallAttempt\_HttpStatusCode
  + awsSdk\_ApiCallAttempt\_HttpStatusCode\_counter\_total
* **Panel Type**: Pie Chart or Table
* **PromQL**:
  + Group by HTTP status code:

promql

Copy code

sum(rate(awsSdk\_ApiCallAttempt\_HttpStatusCode\_counter\_total[1m])) by (status\_code)

**9. Backoff Delays**

* **Purpose**: Monitor delay introduced by retry backoff.
* **Metrics**:
  + awsSdk\_ApiCallAttempt\_BackoffDelayDuration\_seconds
* **Panel Type**: Histogram
* **PromQL**:
  + Distribution of backoff delays:

promql

Copy code

histogram\_quantile(0.95, rate(awsSdk\_ApiCallAttempt\_BackoffDelayDuration\_seconds\_bucket[5m]))

**Dashboard Layout**

1. **Row 1: Performance Overview**:
   * **API Call Duration** (Line Chart)
   * **Credentials Fetch Duration** (Line Chart)
   * **Endpoint Resolution Duration** (Line Chart)
2. **Row 2: Reliability Indicators**:
   * **Retry Count** (Bar Chart)
   * **HTTP Status Codes** (Pie Chart)
3. **Row 3: Latency Analysis**:
   * **Service Call Duration** (Line Chart)
   * **Time to First Byte** (Line Chart)
   * **Backoff Delays** (Histogram)
4. **Row 4: Marshalling and Debugging**:
   * **Marshalling Duration** (Bar Chart)

**Setting Up the Dashboard**

1. **Data Source**:
   * Ensure your Grafana is connected to a Prometheus data source collecting these metrics.
2. **Panel Configuration**:
   * Use the PromQL queries provided above for each panel.
   * Apply transformations (e.g., Group By or Aggregation) to improve clarity.
3. **Variables**:
   * Create variables for S3 bucket names or API operation types to filter metrics dynamically.
4. **Thresholds and Alerts**:
   * Configure thresholds for critical metrics (e.g., API call duration, retry count) to trigger alerts when exceeding acceptable limits.

This setup provides a comprehensive view of S3 performance and reliability using AWS SDK metrics, enabling better monitoring and debugging. Let me know if you'd like further customization or specific PromQL adjustments!

* 1. **Identify key metrics:** Determine the most relevant metrics for monitoring S3 bucket performance and health based on the provided list. Focus on metrics related to API call duration, credential fetching, endpoint resolution, marshalling, retry count, backoff delay, HTTP status code, and service call duration.
  2. **Choose visualization types:** Select suitable Grafana visualization types for each chosen metric. Consider using graphs, gauges, histograms, or tables to effectively display the data.
  3. **Create PromQL queries:** Write PromQL queries to aggregate and filter the selected metrics. This may involve using functions like sum, avg, max, min, rate, and histogram\_quantile. Ensure queries align with the chosen visualization types.
  4. **Arrange and customize:** Arrange the visualizations on the dashboard in a logical and informative manner. Customize titles, labels, legends, and other visual elements to enhance clarity and readability.

Here's a plan for creating a Grafana dashboard for S3 buckets using the provided AWS SDK metrics and PromQL. Since I don't have access to your specific S3 bucket setup and potential naming conventions (which would influence the PromQL queries), I'll provide a generalized dashboard structure and example queries. You'll likely need to adapt these to your environment.

**1. Key Metrics and Visualizations:**

We'll group related metrics into panels for better organization and insight.

* **API Call Performance:**
  + *Metric:* awsSdk\_ApiCall\_ApiCallDuration\_seconds
  + *Visualization:* Histogram (to see the distribution of API call durations) and Graph (to track trends over time). We can use quantiles (e.g., p90, p99) to understand typical and worst-case performance.
  + *Additional Metrics:* awsSdk\_ApiCall\_RetryCount (graph to correlate with duration spikes)
* **Credential Fetching:**
  + *Metric:* awsSdk\_ApiCall\_CredentialsFetchDuration\_seconds
  + *Visualization:* Graph (to monitor trends and identify slowdowns) and Gauge (to display the current/latest fetch duration).
* **Endpoint Resolution:**
  + *Metric:* awsSdk\_ApiCall\_EndpointResolveDuration\_seconds
  + *Visualization:* Graph (similar to credential fetching). Long resolution times can indicate networking issues.
* **Marshalling Performance:**
  + *Metric:* awsSdk\_ApiCall\_MarshallingDuration\_seconds
  + *Visualization:* Graph (to track trends and identify potential inefficiencies in data serialization/deserialization).
* **Request Breakdown (Attempts):**
  + *Metrics:* awsSdk\_ApiCallAttempt\_ServiceCallDuration\_seconds, awsSdk\_ApiCallAttempt\_TimeToFirstByte\_seconds, awsSdk\_ApiCallAttempt\_TimeToLastByte\_seconds, awsSdk\_ApiCallAttempt\_HttpStatusCode
  + *Visualization:* Graph (to visualize the different components of request latency) and a separate panel with a table or heatmap for HTTP status codes and their counts.
* **Retries and Backoff:**
  + *Metrics:* awsSdk\_ApiCall\_RetryCount, awsSdk\_ApiCallAttempt\_BackoffDelayDuration\_seconds
  + *Visualization:* Graphs (to monitor retry rates and backoff durations, which can indicate throttling or other transient errors).

**2. Example PromQL Queries:**

|  |
| --- |
| promql  # API Call Duration (p90)  histogram\_quantile(0.90, sum(rate(awsSdk\_ApiCall\_ApiCallDuration\_seconds\_bucket[5m])) by (le))  # API Call Retry Count (over the last 5 minutes)  sum(increase(awsSdk\_ApiCall\_RetryCount[5m]))  # Credential Fetch Duration (average over 1 minute)  avg(awsSdk\_ApiCall\_CredentialsFetchDuration\_seconds[1m])  # HTTP Status Code 4xx Count (over the last 10 minutes)  sum(increase(awsSdk\_ApiCallAttempt\_HttpStatusCode{code="4xx"}[10m]))  # Service Call Duration (p95 over 5 minutes)  histogram\_quantile(0.95, sum(rate(awsSdk\_ApiCallAttempt\_ServiceCallDuration\_seconds\_bucket[5m])) by (le))  # Example to filter by specific bucket (replace "your-bucket-name")  sum(rate(awsSdk\_ApiCall\_ApiCallDuration\_seconds\_bucket{bucket="your-bucket-name"}[5m])) by (le) |

**3. Dashboard Arrangement and Customization:**

* **Layout:** Arrange panels logically, grouping related metrics together.
* **Titles and Labels:** Use clear and descriptive titles and labels for all panels and axes.
* **Legends:** Ensure legends are easy to understand and distinguish between different series.
* **Units:** Display appropriate units for each metric (e.g., seconds, milliseconds, count).
* **Alerting:** Set up alerts based on critical metrics (e.g., high error rates, prolonged API call durations) to proactively address issues.

**2. Dashboard Design**

The Grafana dashboard will be structured into the following sections:

API Call Performance: Visualize API call durations, counts, and retry rates.

Credential Fetching: Monitor the time taken to fetch credentials.

Endpoint Resolution: Track the time taken to resolve S3 endpoints.

Request Latency Breakdown: Analyze service call duration, signing duration, and time to first/last byte.

HTTP Status Codes: Display the distribution of HTTP status codes.

Retries and Backoff: Monitor retry counts and backoff delays.

**3. PromQL Queries**

Below are example PromQL queries for each metric combination:

**API Call Duration (p90):**

histogram\_quantile(0.90, sum(rate(awsSdk\_ApiCall\_ApiCallDuration\_seconds\_buckets[5m])) by (le))

**API Call Retry Count:**

sum(increase(awsSdk\_ApiCall\_RetryCount[5m]))

**Credential Fetch Duration (p90):**

histogram\_quantile(0.90, sum(rate(awsSdk\_ApiCall\_CredentialsFetchDuration\_seconds\_bucket[5m])) by (le))

**Endpoint Resolve Duration (p90):**

histogram\_quantile(0.90, sum(rate(awsSdk\_ApiCall\_EndpointResolveDuration\_seconds\_bucket[5m])) by (le))

**Service Call Duration (p90):**

histogram\_quantile(0.90, sum(rate(awsSdk\_ApiCallAttempt\_ServiceCallDuration\_seconds\_bucket[5m])) by (le))

**HTTP Status Code Distribution:**

sum(increase(awsSdk\_ApiCallAttempt\_HttpStatusCode[5m])) by (code)

**Retry Count:**

sum(increase(awsSdk\_ApiCall\_RetryCount[5m]))

**Backoff Delay Duration (p90):**

histogram\_quantile(0.90, sum(rate(awsSdk\_ApiCallAttempt\_BackoffDelayDuration