Splunk Observability Dashboard Documentation

Monitoring Azure DevOps VM Deployment Pipelines

# 📌 Table of Contents

1. Overview

2. Objective

3. Data Flow Architecture

4. Telemetry Strategy

- Approach 1: Inline Stage Telemetry

- Approach 2: Standalone Telemetry Pipeline

- Why Both Approaches

5. Telemetry Data Details

6. Integration Details

7. Dashboard Overview

8. Key Metrics and Dimensions

9. Usage Instructions

10. Alerting and Thresholds

11. Maintenance Guidelines

12. Future Enhancements

13. Appendix

# 1. Overview

This documentation explains the Splunk Observability dashboard designed to monitor Azure DevOps VM deployment pipelines. It helps track pipeline health, stage performance, and deployment efficiency using both real-time and post-deployment telemetry.

# 2. Objective

- Monitor each stage of the VM deployment pipelines.  
- Visualize pipeline execution performance.  
- Identify failed or slow-performing stages.  
- Ensure automation health across segments and environments.  
- Enable data-driven decisions for improvement.

# 3. Data Flow Architecture

Azure DevOps Pipelines  
 │  
 [Telemetry Script]  
 ▼  
Azure DevOps REST APIs  
 ▼  
Extract pipeline run, stage, and task details  
 ▼  
Push metrics to:  
 → SignalFx Metrics API (for Splunk Observability)  
 ▼  
Splunk Observability Dashboard

# 6. Integration Details

📡 Azure DevOps REST APIs  
  
Used to extract telemetry data for pipeline runs.  
  
- Base URL: https://dev.azure.com/{org}/{project}/\_apis/  
- Common endpoints:  
 - /pipelines/{pipelineId}/runs  
 - /build/builds/{runId}/timeline  
 - /pipelines/{pipelineId}/stages  
- Authentication: PAT (Personal Access Token) or OAuth  
  
  
📈 SignalFx Metrics API (Splunk Observability)  
  
- URL: https://ingest.us1.signalfx.com/v2/datapoint  
- Sends:  
 - Gauge metrics (duration)  
 - Custom dimensions  
 - Timestamps

# 4. Telemetry Strategy

## Approach 1: Inline Stage Telemetry

Telemetry data is sent to Splunk Observability at the end of every critical stage in the Azure DevOps pipeline.  
This ensures near real-time observability of each deployment run.  
  
- Implemented as inline PowerShell or Bash script in the pipeline stage.  
- Captures:  
 - Start Time, End Time, Duration  
 - Stage Name  
 - Pipeline Name, Run ID, Agent Info  
 - Status (Success/Failure)  
- Metrics pushed immediately to SignalFx via API.

## Approach 2: Standalone Telemetry Pipeline

A separate Azure DevOps pipeline is scheduled or triggered to backfill telemetry data to Splunk.  
This is used when stage-level telemetry is not enabled or for retrospective analysis.  
  
- Scheduled or manually triggered pipeline.  
- Fetches data using Azure DevOps REST API.  
- Aggregates metrics like:  
 - Stage durations  
 - Number of successful/failed runs  
- Pushes data in batch to SignalFx Metrics API.

## Why Both Approaches

Using both inline and standalone telemetry ensures:  
- Real-time monitoring with immediate feedback.  
- Historical and backfill data availability for missed metrics.  
- Improved reliability and data accuracy.

# 5. Telemetry Data Details

The following types of data are captured and pushed as custom metrics:  
  
- Metric Type: Gauge  
- Dimensions:  
 - pipeline\_name  
 - stage\_name  
 - task\_name (if applicable)  
 - status (succeeded, failed)  
 - run\_id  
 - agent\_name  
 - environment\_name  
- Value:  
 - duration (in seconds or milliseconds)  
- Timestamp:  
 - Either stage end time or current UTC time

# 7. Dashboard Overview

The Splunk Observability dashboard includes several charts, single value widgets, and tables to monitor the following:  
  
- Total number of pipeline executions (daily, weekly)  
- Success rate of stages  
- Average and max stage durations  
- Stage-wise breakdown of pipeline runs  
- Filter by pipeline name, environment, and stage  
  
Widgets are configured to allow filtering by dimensions such as environment, pipeline name, and agent.

# 8. Key Metrics and Dimensions

🔢 Metrics Tracked:  
- pipeline.duration.seconds  
- stage.duration.seconds  
- deployment.failure.count  
  
📐 Dimensions:  
- pipeline\_name  
- stage\_name  
- agent\_name  
- status  
- environment  
- run\_id  
- job\_id  
- task\_id

# 9. Usage Instructions

To use the dashboard:  
  
1. Open Splunk Observability Cloud.  
2. Navigate to Dashboards > Azure DevOps Pipeline Monitoring.  
3. Apply filters (Pipeline Name, Environment, Stage Name).  
4. Click on widgets for drill-downs and time-range filtering.  
5. Use split-by or group-by to compare metrics across stages or environments.

# 10. Alerting and Thresholds

Optional alerts can be configured based on:  
  
- Pipeline failures > 3 in an hour  
- Stage duration > 10 minutes  
- Success rate < 90%  
  
Use Detector Rules in Splunk Observability to configure alerts and send to Slack, Email, or Microsoft Teams.

# 11. Maintenance Guidelines

- Validate telemetry script in every pipeline update.  
- Review dashboard filters and widget configs monthly.  
- Clean unused metrics or dimensions to reduce clutter.  
- Audit alerts every quarter.

# 12. Future Enhancements

- Introduce anomaly detection based on pipeline durations.  
- Correlate telemetry with VM provisioning logs.  
- Automate alert tuning using machine learning.  
- Integrate with ServiceNow for automated incident creation.

# 13. Appendix

Sample SignalFx Payload:  
  
{  
 "gauge": [  
 {  
 "metric": "stage.duration.seconds",  
 "value": 134,  
 "timestamp": 1690000000000,  
 "dimensions": {  
 "pipeline\_name": "VM\_Provision",  
 "stage\_name": "Deploy\_VM",  
 "status": "success",  
 "environment": "dev"  
 }  
 }  
 ]  
}