Service Manual for SmartHive

The SmartHive Service Manual provides comprehensive guidance for the setup, operation, and maintenance of the SmartHive platform. It covers various aspects including hardware and software configurations, system architecture, troubleshooting techniques, and best practices for optimal performance and reliability.

By following the guidelines and instructions in this manual, users can maximize the performance and reliability of SmartHive.

Pre-Installation Checks

Before installing the SmartHive platform, it is crucial to perform preinstallation checks to ensure the environment is properly configured for on-prem deployment. The following checks should be performed:

Hardware Requirements

Following is the matrix table for the CPU, Memory, and HDD for each VM based on the provided hardware requirements:

VM	RAM	HDD	CPU
MES Cluster			
HAProxy	8 GB	100 GB	4
Control Box	8 GB	250 GB	4
Kubegres (worker)	16 GB	1000 GB/1TB	4
Kafka (worker)	16 GB	1000 GB/1TB	4
Kubernetes Master	4 GB	200 GB	2
Kubernetes Worker	16 GB	200 GB	8
Druid Cluster			
HAProxy	8 GB	100 GB	4
Control Box	8 GB	250 GB	4
Kubernetes Master	4 GB	200 GB	2
Kubernetes Worker	16 GB	200 GB	8
Kubegres (worker)	16 GB	1000 GB/1TB	4

For the lightweight infrastructure configuration, the recommended specs are:

- HDD: 500GB
- RAM: 24 GB
- CPU: 8

Ubuntu Version

Verify that the Ubuntu version installed on each VM is 22.04.03, as SmartHive platform is designed to run on this version.

Python Version

Verify that the Python version installed on each VM is 22.04.2, as SmartHive platform is designed to run on this version.

Prerequisites Installation and Configuration

Confirm that all prerequisites required for the SmartHive platform are installed and configured correctly on each VM. This includes:

- **Required packages:** Ensure that all required packages are installed.
- **Password-less sudo setup:** Verify that password-less sudo access is configured correctly for authorized users.
- **SSH setup for NCL execution:** Ensure that SSH is properly configured to allow secure communication between nodes. Before proceeding with the installation of the SmartHive platform, it is essential to perform a series of preinstallation checks to ensure that the environment is properly configured and ready for deployment. The following checks should be performed.

Environments

The SmartHive platform is designed to be deployed in three main environments: development, QA, and production. Each environment is hosted on-premises with Ubuntu 22.04.03 VMs. The development environment is smaller in scale compared to QA and production, which are identical in configuration to ensure consistency across environments. These environments are crucial for ensuring the quality, reliability, and scalability of the platform.

Development Environment

The development environment is dedicated to the initial development and testing of new features and functionalities. It provides a sandbox-like environment for developers to experiment without affecting the production environment.

Tools/Technologies

The development environment is built on Ubuntu 22.04.03 and utilizes virtual machines (VMs) for hosting. Docker and Kubernetes are used for containerization and orchestration, enabling efficient deployment and management of applications.

Configuration

The development environment consists of Ubuntu VMs, each with minimal worker nodes to support development activities. This environment is designed to be flexible and easily configurable to accommodate rapid changes during the development phase.

QA Environment

The QA (Quality Assurance) environment is a crucial component of the software development lifecycle, serving as a dedicated space for testing features and functionalities before they are deployed to the production environment. It plays a vital role in ensuring the quality, reliability, and performance of the SmartHive platform.

Tools/Technologies

The QA environment utilizes the same tools and technologies as the development environment to maintain consistency and facilitate seamless testing. These tools include Ubuntu 22.04.03, virtual machines (VMs), Docker, and Kubernetes. By using the same environment setup as development, QA teams can accurately replicate testing scenarios and identify potential issues early in the development process.

Configuration

The QA environment consists of Ubuntu VMs configured to mirror the production environment as closely as possible. This configuration ensures that the testing environment accurately reflects the conditions under which the platform will operate in production, allowing for more accurate and reliable testing results.

- **Identical Configuration:** Each Ubuntu VM in the QA environment is configured identically to the production environment, including the same software versions, configurations, and settings. This ensures that any issues identified in QA can be reliably reproduced and addressed before deployment to production.
- **High Availability Setup:** The QA environment is designed with a high availability setup to ensure continuous testing even in the event of hardware or software failures. This ensures that testing can continue uninterrupted and that any issues can be identified and resolved quickly.
- **Load Balancing:** Load balancing is implemented in the QA environment to distribute traffic evenly across the VMs. This helps simulate real-world usage scenarios and ensures that the platform can handle varying levels of traffic and usage patterns.

- **Testing Scenarios:** The QA environment is used to test various scenarios, including functional testing, performance testing, and compatibility testing. This ensures that the platform meets the specified requirements and performs as expected under different conditions.

Production Environment

The production environment is the live deployment environment for the SmartHive platform, where the final version of the software is hosted and made available to end users. It is designed to be stable, secure, and highly available to ensure uninterrupted service and optimal performance for users.

Tools/Technologies

The production environment utilizes the same tools and technologies as the development and QA environments to maintain consistency and facilitate seamless deployment. These tools include Ubuntu 22.04.03, virtual machines (VMs), Docker, and Kubernetes. By using the same environment setup across all stages, the production environment can accurately reflect the conditions tested in QA, minimizing the risk of issues arising in production.

Configuration

The production environment is configured to ensure high availability and optimal performance for the SmartHive platform. The key components of the production environment configuration include:

- **Ubuntu VMs:** The production environment consists of Ubuntu VMs, each hosting a part of the platform. This setup allows for load distribution and redundancy, ensuring that the platform remains accessible even in the event of hardware or software failures. To provide High Availability, you must have a minimum of 3 masters.
- **High Availability Setup:** The production environment is configured with a high availability setup to minimize downtime and ensure continuous service availability. This includes redundant hardware, failover mechanisms, and load balancing to distribute traffic across the VMs.
- **Load Balancing:** Load balancing is implemented in the production environment to distribute incoming traffic evenly across the VMs. This ensures optimal performance and prevents any single VM from being overloaded, leading to better user experience.
- **Security Measures:** The production environment is secured using various security measures, including firewalls, encryption, and access control mechanisms. These measures help protect the platform from unauthorized access, data breaches, and other security threats.
- **Monitoring and Logging:** The production environment is equipped with monitoring and logging tools to track performance metrics, detect anomalies, and troubleshoot issues in real-time. This helps ensure the stability and reliability of the platform.

Maintenance and Updates

Regular maintenance and updates are essential for ensuring the security and performance of the production environment. This includes applying security patches, updating software components, and performing regular backups to prevent data loss.

Keycloak Security Environment

Keycloak is a powerful open-source identity and access management (IAM) solution that provides centralized security and user management capabilities. Here are some key aspects of a Keycloak security environment:

Supported Protocols

1. Keycloak supports industry-standard security protocols such as **OAuth 2.0**, **OpenID Connect**, and **SAML**.
2. As an OAuth2, OpenID Connect, and SAML compliant server, Keycloak can secure any application or service as long as the technology stack they are using supports any of these protocols.

Secure Applications and Services

Follow these steps to secure an application or service using Keycloak:

1. Register a client to a realm using the Keycloak Admin Console, the client registration service, or the CLI.
2. Enable either **OpenID Connect** or **SAML** protocols in your application:
   - Leverage existing OpenID Connect and SAML support from the application ecosystem.
   - Alternatively, use a Keycloak Adapter if needed.

> **Note**: Detailed instructions for these steps can be found in the Server Administration Guide.

Getting Started with Keycloak

1. The Keycloak Quickstarts Repository provides examples for securing applications and services using different programming languages and frameworks.
2. By exploring the documentation and codebase, you'll understand the minimal changes required in your application to secure it with Keycloak.
3. Additionally, there are trusted and well-known client-side implementations for both OpenID Connect and SAML protocols.

Benefits of Keycloak

1. Keycloak acts as a centralized identity provider, allowing organizations to manage user identities, define access policies, and authenticate users across multiple applications and services.
2. Keycloak offers Single Sign-On (SSO), meaning users only need to log in once to access multiple applications.

Kubernetes Monitoring Environment

Kubernetes monitoring helps you identify issues and proactively manage your Kubernetes clusters. By tracking key metrics, you can ensure optimal resource utilization, uptime, and overall cluster health. Here are some essential aspects of Kubernetes monitoring:

Measurable Metrics:

- **Cluster Monitoring**: Keep track of the health of the entire Kubernetes cluster. Verify if nodes are functioning properly, their capacity, and how resources are utilized across the cluster.
- **Pod Monitoring**: Monitor individual pods for resource utilization, application metrics, and replication/auto-scaling-related metrics.
- **Deployment Metrics**: When using Prometheus, monitor Kubernetes deployments, including cluster CPU, Kube state, cAdvisor, and memory metrics.
- **Ingress Metrics**: Monitor ingress traffic to identify and manage issues related to workload health and network traffic.
- **Persistent Storage**: Set up monitoring for volume health to ensure proper implementation of Container Storage Interface (CSI).
- **Control Plane Metrics**: Monitor schedulers, API servers, and controllers to visualize cluster performance for troubleshooting.
- **Node Metrics**: Monitor CPU and memory for each Kubernetes node to prevent resource exhaustion.

Kubernetes Monitoring Tools

- **Kubernetes Dashboard**: Provides a web-based UI for managing and monitoring Kubernetes clusters.
- **Prometheus**: A powerful open-source monitoring and alerting toolkit designed for cloud-native environments.
- **Grafana**: Integrates with Prometheus to create customizable dashboards for visualizing metrics.
- **EFK Stack (Elasticsearch, Fluentd, Kibana)**: Used for log aggregation and analysis.

- **LOKI**: A lightweight log aggregation system that works well with Kubernetes.

Best Practices for Kubernetes Monitoring

- **Single Pane of Glass**: Use a unified dashboard or tool to monitor all relevant metrics in one place.
- **Scalability and Data Retention**: Ensure your monitoring systems can scale as your cluster grows and retain sufficient historical data for analysis.
- **Alerting and Notification**: Set up alerts for critical issues and deliver them to the appropriate staff members.
- **CI/CD Integration**: Integrate monitoring systems with your continuous integration and continuous deployment (CI/CD) pipeline for seamless monitoring during application updates.

ELK Stack Environment

The ELK Stack is a powerful combination of open-source tools for managing and analyzing data. It provides a comprehensive solution for managing large volumes of data, offering real-time insights, analytics, and visualization. The ELK Stack consists of the following components:

Elasticsearch

Elasticsearch is a distributed, JSON-based search and analytics engine. It allows you to store, search, and analyze data with speed at scale.

Use Cases: Whether you're looking for specific actions from an IP address, analyzing spikes in transaction requests, or even hunting for a nearby taco spot, Elasticsearch provides the foundation for efficient search and analysis.

Logstash

Logstash is a data processing pipeline that ingests, transforms, and enriches data from various sources. It allows you to collect logs, metrics, and other data, and then send it to Elasticsearch for indexing and analysis.

Use Cases: Logstash is commonly used for log aggregation, data enrichment, and data transformation before it reaches Elasticsearch.

Kibana

Kibana is the extensible user interface for Elasticsearch. It gives shape to your data by providing stunning visualizations, dashboards, and real-time insights.

Use Cases:

- Explore data with visualizations like waffle charts, heatmaps, and time series analysis.
- Create preconfigured dashboards for diverse data sources.
- Highlight key performance indicators (KPIs) in live presentations.
- Manage your deployment within a single UI.

DNS Setup Environment

DNS (Domain Name System) environment is essential for translating human-friendly domain names into IP addresses that computers can understand.

> **Note**: Make sure that the VIP and all IP addresses are static including VIP. The DNS should point to the VIP and it should not be changed.

Jenkins Environment

Environment variables play a crucial role in configuring and customizing Jenkins pipelines. They allow you to set global or stage-specific values that can be accessed within your Jenkins file. Here are some key points about Jenkins environment variables:

Global Environment Variables

You can define environment variables globally, making them accessible across all stages of your pipeline. For example, in a Jenkins file (Declarative Pipeline), you can set global environment variables like this:

```
pipeline {
    agent {
        label '!windows'
    }
    environment {
        DISABLE_AUTH = 'true'
        DB_ENGINE = 'sqlite'
    }
    stages {
        stage('Build') {
            steps {
                echo "Database engine is ${DB_ENGINE}"
                echo "DISABLE_AUTH is ${DISABLE_AUTH}"
                sh 'printenv'
            }
        }
    }
}
```

Note: In this example, `DISABLE_AUTH` and `DB_ENGINE` are global environment variables that can be accessed throughout the pipeline.

Local Environment Variables

You can also define environment variables at the specific stage level or inside the script block. These local variables are scoped to the respective stage or block where they are defined. For more advanced use cases, you can toggle to a Scripted Pipeline and define environment variables within the script.

Benefits of Using Environment Variables

- **Code Reusability**: Instead of hardcoding values, use environment variables to avoid repetition and make your pipeline more maintainable.
- **Security**: Avoid putting sensitive information directly into the Jenkins file. Use environment variables to securely manage credentials and other secrets.
- **Flexibility**: Change environment variables without modifying the pipeline code itself.
- **Integration with Plugins**: Some Jenkins plugins may also set environment variables. Refer to plugin documentation.

Druid Database Environment

Apache Druid is a high-performance, real-time analytics database that empowers organizations to process massive amounts of data with sub-second query response times. Apache Druid's environment combines scalability, real-time capabilities, and efficient query execution to unlock the full potential of data analytics. Here are some key aspects of Druid's environment:

Datasources

In Druid, data is organized into datasources. Think of datasources as analogous to tables in relational databases. A Druid cluster can handle multiple datasources concurrently, each ingested from various sources. By default, Druid partitions data based on time, but you can further partition it based on other attributes if needed.

Cluster Architecture

Druid operates in a clustered environment, where data is distributed across different machines within the system.

The three primary server types in a Druid cluster are:

- **Master**: Responsible for managing metadata, task coordination, and query distribution.
- **Query**: Handles incoming queries and executes them against the data.
- **Data**: Stores ingested data and serves query results.

The data nodes (Data servers) collectively form the deep storage layer, which stores all ingested data. This deep storage is accessible by every Druid server. In a clustered deployment, deep storage is typically backed by distributed object stores like Amazon S3 or HDFS. In a single-server setup, local disk storage is commonly used.

Query Execution and Performance

Druid's query engine executes OLAP queries in milliseconds, even on high-cardinality and high-dimensional datasets with billions to trillions of rows. Scatter/gather techniques are employed to achieve high-speed queries. Data is preloaded into memory or local storage, minimizing data movement and network latency. Druid's tiering and quality of service (QoS) configuration ensures optimal price-performance for mixed workloads, avoiding resource contention. The architecture allows for true stream ingestion, query-on-arrival, and low-latency ingestion, making it ideal for real-time and historical insights.

SQL Support

Developers and analysts can leverage familiar SQL for end-to-end data operations within Druid. Whether it's ingesting, transforming, or querying data, Druid provides a seamless SQL interface.

Postgres Environment

PostgreSQL or Postgres is a powerful open-source relational database management system. It is widely used for various applications, from small-scale projects to large enterprise systems.

Environment Variables

Postgres uses environment variables to configure connection parameters. These variables allow you to avoid hard-coding database connection information directly into your client applications.

Here are some important environment variables:

1. `PGHOST`: It refers to the hostname or IP address of the machine where the PostgreSQL server is running.
2. `PGHOSTADDR`: It allows you to specify the IP address (instead of a hostname) of the machine where the PostgreSQL server is running. It is useful for avoiding DNS lookup overhead.
3. `PGPORT`: It specifies the TCP/IP port or local Unix domain socket file extension on which the PostgreSQL server listens for connections from client applications.
4. `PGDATABASE`: It refers to the name of the database that a client application should connect to when establishing a connection with the PostgreSQL server.
5. `PGUSER`: It refers to the user connection parameter when establishing a connection to a PostgreSQL database.
6. `PGPASSWORD`: This environment variable is used to provide a password for password-based authentication (though not recommended for security reasons).
7. `PGPASSFILE`: This is an environment variable in PostgreSQL that specifies the location of the password file.

8. `PGSERVICE`: This environment variable in PostgreSQL allows libpq connection parameters to be associated with a single service name. This service name can then be specified in a libpq connection string, and the associated settings will be used.
9. `PGSERVICEFILE`: It specifies the per-user connection service file (usually `~/.pg_service.conf` or `%APPDATA%\\postgresql\\.pg_service.conf` on Windows).
10. `PGSSLMODE`: This environment variable in PostgreSQL controls the level of protection provided for PostgreSQL SSL connections.
11. `PGSSLCERT`: This environment variable in PostgreSQL specifies the location of the SSL certificate file to be used for SSL connections.

These environment variables can be set in your shell or in a script before you run a PostgreSQL client application. These variables provide a flexible way to manage the PostgreSQL connections and settings.

> **Note**: Using environment variables for sensitive information like passwords is not recommended due to security risks. Instead, consider using other methods such as password files or interactive prompts for authentication.

Setting of Environment Variables

If you're using PostgreSQL on Windows, you can set the environment variables by following these steps:

1. Type "environment" in the Windows search bar and select "Edit environment variables for your account."
2. Under the "System Variables" section, find the "PATH" variable and click "Edit."
3. Add the path to your PostgreSQL installation directory to the list of paths.

Superset Environment

Apache Superset is an open-source data exploration and visualization platform that allows you to create interactive dashboards, charts, and reports. Here's how you can get started with Superset:

Kafka Environment

A Kafka environment typically refers to the infrastructure and ecosystem surrounding Apache Kafka, an open-source distributed event streaming platform. In a Kafka environment, several components work together to enable the ingestion, storage, processing, and consumption of data streams. These components may include Kafka brokers, which manage the storage and replication of data across clusters; ZooKeeper, which coordinates and manages Kafka brokers; producers, which publish data to Kafka topics; consumers, which subscribe to topics and process data; and various tools for monitoring, management, and data integration.

The Kafka environment is designed to handle high volumes of data with low latency, making it suitable for use cases such as real-time analytics, event sourcing, log aggregation, and stream processing. Organizations often deploy Kafka environments to build scalable and resilient data pipelines that facilitate real-time data processing, event-driven architectures, and integration between different systems and applications. Proper configuration, monitoring, and management of the Kafka environment are essential to ensure optimal performance, reliability, and security.

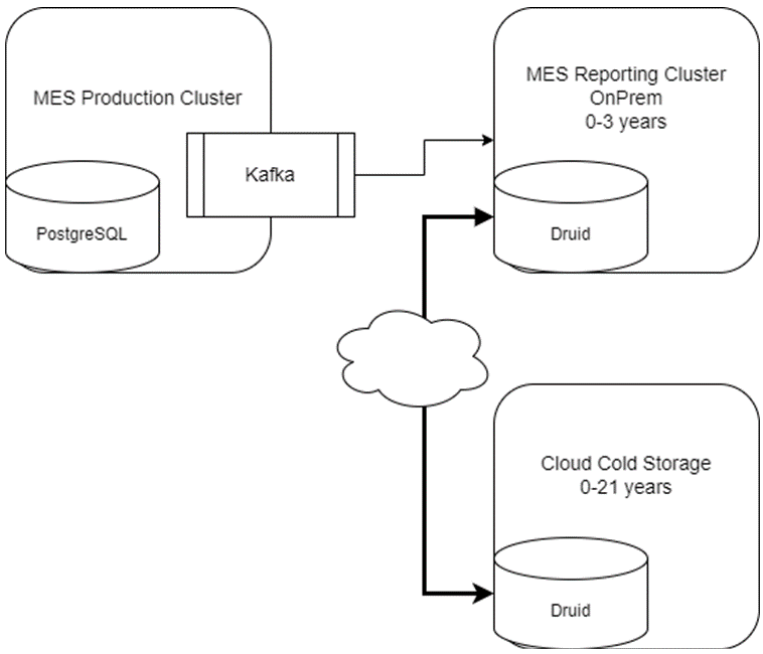
These environments interact with each other through APIs, data pipelines, network communication, and integration points to provide a cohesive ecosystem for developing, deploying, monitoring, and analyzing applications and services in an enterprise environment. Integration between these environments enables seamless communication, data exchange, and collaboration across different components of the system.

Communication and Data Exchange

Interacting with Kafka

The Manufacturing Execution System (MES) utilizes Kafka for data streaming and Druid for data storage and analytics. The architecture ensures that recent data is readily available for immediate reporting and analysis on-premises. In contrast, historical data is stored in the cloud for long-term retention and occasional access. This setup leverages Kafka for efficient real-time data streaming and Druid for powerful data storage and analytics.

The following diagram shows a data architecture for MES that utilizes Kafka for data streaming and Druid for data storage and analytics.



Data Sync between the Clusters:

1. Any data written (insert/update) into PostgreSQL database is intercepted and published to Kafka topic. Any machine raw data read by IF adapters are published to Kafka topics directly without storing them in PostgreSQL database.
2. Kafka streams data to the MES Reporting Cluster (OnPrem), where Druid stores recent data (0-3 years) for quick access and analytics. OnPrem is an analytics data store designed for fast aggregations and queries on large datasets.

> **Note**: Druid OnPrem consumes the topic, and the data is indexed and written into Druid database. Kafka default retention is 7 days. Even if there are issues in Druid consumption, it can still consume within 7 days. The retention is configurable to keep more days.

3. Kafka also streams data to Cloud Cold Storage, where Druid stores long-term data (0-21 years) for historical analysis. Cloud Cold Storage provides a scalable and cost-effective solution for historical data storage and analysis.

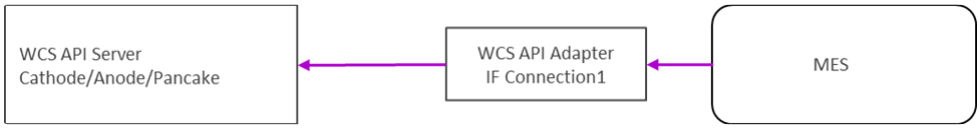
> **Note**: The indexing duration is anywhere between 1 min to 24 hours. Once the indexing duration is completed, the data segment from On-prem is pushed to cloud based on the protocol supported by Cloud provider. For instance, S3 for AWS and Blob for Azure.

Interacting with Rest API

The Warehouse Control System (WCS) is spread out to 3 different servers at Cathode, Anode and Pancake warehouses. MES has to provide the Master data and few requests in the form of Rest APIs. WCS provides various contracts like Cycle counts, Operational logs and Equipment performances in the form of CSV files without headers.

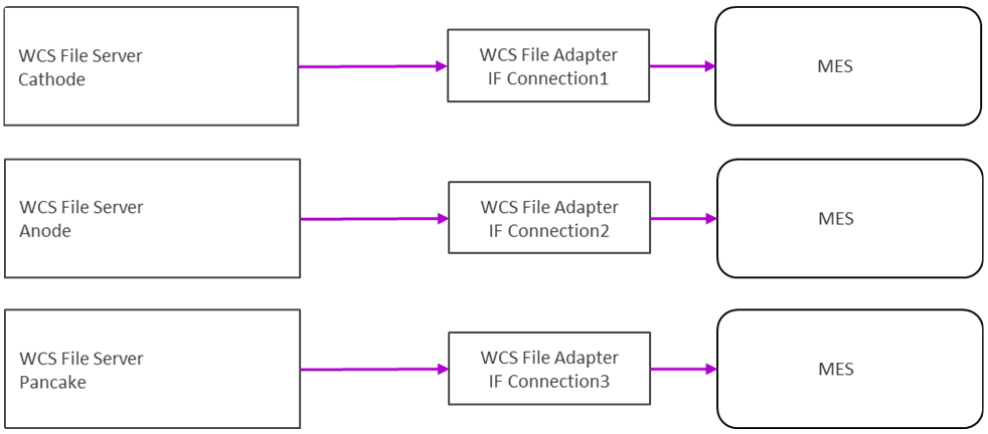
All the CSV files are shared in 1 folder by each warehouse. That means all the Cathode files are in 1 folder in one server, Anode files are in 1 folder in another server and same for Pancake warehouse.

The following diagram shows a data interchange, ensuring that the MES can access and respond to data managed by the WCS API Server through the WCS API Adapter (IF Connection1). It helps in streamlining manufacturing processes and improving operational efficiency.



- WCS API Server handles API requests related to specific items or processes (Cathode, Anode, Pancake).
- WCS API Adapter (IF Connection1) acts as an intermediary that facilitates communication between the WCS API Server and the MES.
- MES provides data to the WCS through the API Adapter.

The following diagram shows a structured setup where each file server has its dedicated adapter connection to the MES. This ensures organized and streamlined data processing from various parts of the WCS to the MES, enhancing efficiency and reliability in the manufacturing execution process.



- Each WCS File Server communicates with the MES through a dedicated file adapter:
  - WCS File Adapter IF Connection1 for Cathode
  - WCS File Adapter IF Connection2 for Anode
  - WCS File Adapter IF Connection3 for Pancake
- The MES system receives data from each of the WCS File Servers through their respective file adapters.

Kafka Topics and Rest API

Settings for the Kafka Topics

To set up Kafka topics effectively for the infrastructure, you must consider several configurations and settings based on your specific requirements. By following the Kafka settings, you can ensure your Kafka topics are configured for optimal performance, reliability, and scalability.

Follow these steps to set up the Kafka topics:

1. Choose a meaningful and descriptive name for the topic.
   Example: `mes_production_data`, `mes_reports_data`.
2. Determine the number of partitions based on the expected throughput and parallelism.
   A higher partition count allows higher throughput but increases complexity. Example: `12` partitions for high throughput environments.
3. Set replication factor to ensure data redundancy and high availability.
   Typically set to `3` in a production environment. Ensure it is not higher than the number of brokers.
4. Define retention period for how long Kafka retains the messages.
   Example: `retention.ms=604800000` (one week). For long-term storage, configure your sink (like a database or data lake).
5. Choose one of the following cleanup policies (Example: `cleanup.policy=delete`):
   - `delete`: Deletes old messages after the retention period.
   - `compact`: Keeps the latest version of each key.
6. Set Minimum In-Sync Replicas (ISR) to ensure a minimum number of replicas that must acknowledge a write for it to be considered successful. Example: `min.insync.replicas=2`.
7. Set the segment size to control the size of log segments.
   Smaller segments can make log compaction more efficient. Example: `segment.bytes=1073741824` (1GB).
8. Set the maximum message size. Example: `max.message.bytes=1048576` (1MB).
9. Use compression to save disk space and network bandwidth.
   The common options: `gzip`, `snappy`, and `lz4`. Example: `compression.type=snappy`.
10. Control the maximum lag allowed for a follower before it is removed from the ISR.
    Example: `replica.lag.max.messages=4000`.

Configuration of Kafka Topic

The following is an example of a Kafka topic configuration in a YAML format for a topic named `mes_production_data`:

```
topics:
- name: mes_production_data
num_partitions: 12
replication_factor: 3
config:
retention.ms: 604800000 # 7 days
cleanup.policy: delete
min.insync.replicas: 2
segment.bytes: 1073741824 # 1GB
max.message.bytes: 1048576 # 1MB
compression.type: snappy
replica.lag.max.messages: 4000
```

Creating Kafka Topics in Bulk

Follow these steps to create topics in bulk:

1. Create a file named "topics" and add the names of the topics that need to be created.
2. Create a topic manifest file named "topics.yml.orig" with the following contents:

```
---
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaTopic
metadata:
 name: #TOPIC#
 namespace: "sm-kafka"
 labels:
   strimzi.io/cluster: "sm-main-kafka"
spec:
 partitions: 10
 replicas: 3
```

3. Create a file named "add-topic.sh" with the following contents and make it executable.

```
#!/bin/bash
TOPICS=$(cat topics)
for TOPIC in $TOPICS
do
  cp topic.yml.orig topic.yml
  sed -i "s/#TOPIC#/$TOPIC/" topic.yml
  kubectl apply -f topic.yml
done
```

> **Note**: You can use the following command to delete the Kafka topic: `kubectl delete kt <topic name> -n <namespace>`.

Settings for the Rest API

Configuring a REST API involves several key settings to ensure it operates securely, efficiently, and reliably.

Follow these steps to set up the REST API:

Basic Configuration:

1. Define the base URL for the API.
   Example: `https://api.example.com/v1/`
2. Define the endpoints for the different resources.
   Example: `/users`, `/orders`, `/products`.
3. Use appropriate HTTP methods for operations.
   - `GET` for retrieving data.
   - `POST` for creating data.
   - `PUT` for updating data.

- `DELETE` for deleting data.

4. Use JSON for request and response bodies.
   Example: `Content-Type: application/json`.
5. Implement versioning in the API URL or headers.
   Example: `https://api.example.com/v1/` or `Accept: application/vnd.example.v1+json`.

Security Settings:

1. Use OAuth 2.0, JWT, or API keys for authentication.
   Example: `Authorization: Bearer <token>`.
2. Implement role-based access control (RBAC) for authorization.
   Define user roles and permissions for accessing different endpoints.
3. Enforce HTTPS to secure data in transit.
   Obtain and install SSL/TLS certificates.
4. Configure Cross-Origin Resource Sharing (CORS) to allow or restrict requests from other domains.
   Example:
   json

   `{ "Access-Control-Allow-Origin": "*", "Access-Control-Allow-Methods": "GET, POST, PUT, DELETE", "Access-Control-Allow-Headers": "Content-Type, Authorization" }`

Performance Settings:

1. Implement rate limiting to prevent abuse.
   Example: Allow 100 requests per minute per user.
2. Use caching to improve response times. Cache static content and frequently accessed data.
   Example: `Cache-Control: max-age=3600`.
3. Enable gzip or Brotli compression for responses to reduce payload size.
   Example: `Content-Encoding: gzip`.

Error Handling:

1. Define a standard format for error responses.
   Example:

   `{ "error": { "code": 404, "message": "Resource not found" } }`
2. Use appropriate HTTP status codes for responses.
   - `200 OK` for successful requests.
   - `201 Created` for successful resource creation.
   - `400 Bad Request` for invalid requests.
   - `401 Unauthorized` for authentication errors.
   - `403 Forbidden` for authorization errors.
   - `404 Not Found` for missing resources.
   - `500 Internal Server Error` for server errors.

Configuration of REST API

The following is an example of setting up a REST API using Node.js and Express:

```
const express = require('express');
const bodyParser = require('body-parser');
const helmet = require('helmet');
const cors = require('cors');
const jwt = require('jsonwebtoken');
```

```
const app = express();
const PORT = process.env.PORT || 3000;

// Middleware
app.use(bodyParser.json());
app.use(helmet());
app.use(cors());

// Mock database
const users = [];

// Authentication middleware
const authenticateToken = (req, res, next) => {
const authHeader = req.headers['authorization'];
const token = authHeader && authHeader.split(' ')[1];
if (token == null) return res.sendStatus(401);
jwt.verify(token, process.env.ACCESS_TOKEN_SECRET, (err, user) => {
if (err) return res.sendStatus(403);
req.user = user;
next();
});
};

// Endpoints
app.get('/users', authenticateToken, (req, res) => {
res.json(users);
});
app.post('/users', (req, res) => { const user = req.body;
users.push(user);
res.status(201).json(user);
});

// Error handling
app.use((err, req, res, next) => {
console.error(err.stack);
res.status(500).json({ error: 'Something went wrong!' });
});

// Start server
app.listen(PORT, () => {
console.log(`Server is running on port ${PORT}`);
});
```

Configuration Management

A `config.ini` file is used to store configuration settings for software applications. It's a plain text file that follows a standard format to define various settings and parameters the software needs to function correctly. The `config.ini` file provides a structured approach to configure various software components within the MES and Kafka ecosystem.

Example `config.ini`

```
[Kafka]
bootstrap.servers=localhost:9092
topic.name=mes-data
replication.factor=3
partitions=10
acks=all
retries=3

[PostgreSQL]
host=localhost
port=5432
database=mes_production
user=mes_user
password=mes_password
max.connections=100

[Druid]
host=localhost
port=8082
database=mes_reporting
retention.period=3y
max.connections=50

[WCSAPI]
server.url=http://wcs-api-server
api.version=v1 timeout=30s
auth.token=securetoken123

[MES]
endpoint=http://mes-server
auth.token=securetoken123
connection.type=IF_Connection1
file.transfer.timeout=60s

[Kubernetes]
master.nodes=3
worker.nodes=5
api.server.port=6443
dashboard.port=443
ui.port=80 reporting.port=9092

[HAProxy]
config.file=/etc/haproxy/haproxy.cfg
restart.command=systemctl
restart haproxy
```

`[VirtualIPs]`

`production.vip=https://shmedb.mesproduction.com`

`reporting.vip=https://shmebd.mesreports.com`

Software Configuration, Purpose, and Use

Kafka

- **Purpose:** Configures the Kafka cluster for data streaming and message brokering.
- **Use:**
  - `bootstrap.servers`: Specifies the Kafka broker addresses.
  - `topic.name`: Defines the name of the Kafka topic.
  - `replication.factor`: Sets the replication factor for fault tolerance.
  - `partitions`: Determines the number of partitions for parallel processing.
  - `acks`: Ensures message durability (e.g., `all` waits for full acknowledgment).
  - `retries`: Specifies the number of retry attempts for failed message sends.

PostgreSQL

- **Purpose:** Sets up the connection details for the PostgreSQL database used in the MES Production Cluster.
- **Use:**
  - `host`, `port`: Define the database server's location and port.
  - `database`, `user`, `password`: Provide the credentials for database access.
  - `max.connections`: Limits the maximum number of concurrent connections.

Druid

- **Purpose:** Configures the Druid database for analytical data storage in the MES Reporting Cluster.
- **Use:**
  - `host`, `port`: Set the Druid server's address and port.
  - `database`: Names the database used for reporting.
  - `retention.period`: Determines how long data is retained.
  - `max.connections`: Limits the number of concurrent connections.

WCSAPI

- **Purpose:** Manages the configuration for the WCS API Server, facilitating communication between MES and WCS.
- **Use:**
  - `server.url`: Specifies the URL of the WCS API Server.
  - `api.version`: Indicates the API version to use.
  - `timeout`: Sets the request timeout duration.
  - `auth.token`: Provides the authentication token for secure communication.

MES

- **Purpose:** Configures the MES (Manufacturing Execution System) endpoint and connection settings.
- **Use:**
  - `endpoint`: Defines the MES server's URL.
  - `auth.token`: Supplies the authentication token for MES access.
  - `connection.type`: Specifies the type of connection to use (e.g., `IF_Connection1`).
  - `file.transfer.timeout`: Sets the timeout duration for file transfers.

Kubernetes

- **Purpose:** Details the configuration for Kubernetes clusters managing the MES and Reporting environments.
- **Use:**
  - `master.nodes`: Number of master nodes in the cluster.
  - `worker.nodes`: Number of worker nodes.
  - `api.server.port`: Port for accessing the Kubernetes API server.
  - `dashboard.port`, `ui.port`: Ports for accessing the Kubernetes dashboard and MES UI.
  - `reporting.port`: Port for reporting services.

HAProxy

- **Purpose:** Configures the HAProxy load balancer for high availability and traffic management.
- **Use:**
  - `config.file`: Path to the HAProxy configuration file.
  - `restart.command`: Command to restart the HAProxy service after configuration changes.

VirtualIPs

- **Purpose:** Defines the virtual IP addresses for accessing the MES UI and reporting services.
- **Use:**
  - `production.vip`: URL for the production MES UI.
  - `reporting.vip`: URL for the MES reporting services.

Infrastructure Components

The following table displays the infrastructure components along with their respective versions and descriptions:

Software	Version	Description
Ubuntu	22.04.03	OS component
Kubernetes	1.24	Open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications
Kubegres	1.15	Containerized PostgreSQL with which we can scale and achieve the High Availability
Kubekafka	2.8	Containerized Apache Kafka with which we can scale and achieve the High Availability
Keycloak	19.0.1	Identity Server by managing users and passwords in Encrypted format
MetalLB	v0.12.1	Software load balancer designed for bare metal Kubernetes clusters
Ingress	1.2.0	API routing module to external client requests to internal applications
Redis	7.0.11-debian-11-r12	Cache module used for caching frequently used data for better performance
Grafana (for SH and Druid)	10.2.3	Open-source platform for monitoring and observability, designed to visualize and analyze metrics and other data from various sources. It provides a web-based interface for creating, exploring, and sharing interactive dashboards and panels
Prometheus	2.48.1	Open-source platform for monitoring and observability, designed to visualize and analyze metrics and other data from various sources. It

		provides a web-based interface for creating, exploring, and sharing interactive dashboards and panels
ElasticSearch (ELK)	7.9.0	Stands for Elasticsearch, Logstash, and Kibana. These 3 tools provide a powerful stack used for log and data analysis
Superset	2.1.0	Open-source data exploration and visualization platform that facilitates the creation and sharing of interactive dashboards
OpenEBS	3.4.0	Open-source container-attached storage platform for Kubernetes. It provides a way to manage and orchestrate persistent storage for stateful applications running in Kubernetes environments
Istio	1.15.0	Istio is an open-source service mesh platform designed to connect, manage, and secure microservices within a containerized application environment
Secrets replicator	2.8.0	A software module to replicate the application secrets across all its replicas/pods
Helm	3.7.1	Open-source package manager for Kubernetes applications
NussKnacker (Planned for the future)	1.11.3	Open-source stream processing and Complex Event Processing (CEP) platform. It provides a visual and user-friendly interface for designing, deploying, and managing stream processing applications.
Valero	1.12.3	An open-source tool to safely backup and restore, perform disaster recovery, and

		migrate Kubernetes cluster resources and persistent volumes
Druid	0.22.1	Open-source, column-oriented, distributed data store designed for real-time analytical queries on large datasets

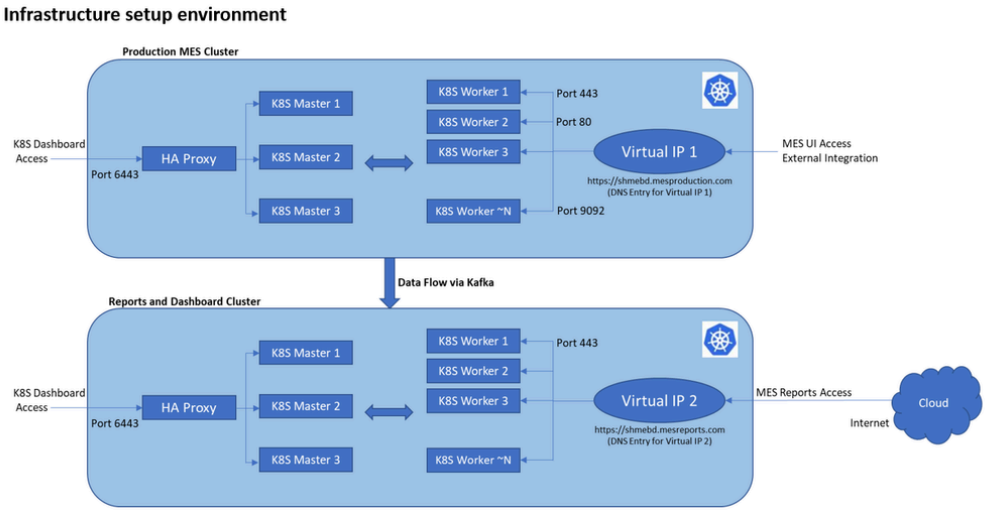
Hybrid Infrastructure

Kafka facilitates the data flow between the following Kubernetes (K8S) clusters:

- Production MES Cluster
- Reports and Dashboard Cluster

This infrastructure setup provides a robust environment for managing MES production and reporting, ensuring high availability, efficient data flow, and secure access points.

> The hybrid infrastructure supports High Availability. The backup and restore functions are on the cloud. Both MES and Druid clusters are on-prem.



Production MES Cluster:

- **K8S Masters and Workers**:
  - **K8S Master 1, Master 2, and Master 3**: These nodes manage the orchestration and administration of Kubernetes (K8S) worker nodes.
  - **K8S Workers**: Multiple worker nodes (Worker 1, Worker 2, Worker 3, and possibly more) are responsible for running the actual applications and workloads.
- **HA Proxy**: Provides load balancing to distribute the incoming traffic across the K8S masters for high availability. It is accessed through port 6443.
- **Virtual IP 1**: This is the external entry point for the Production MES Cluster. It can be accessed through **Port 443** for HTTPS traffic, **Port 80** for HTTP traffic, and **Port 9092** likely for Kafka broker access. The DNS entry for this virtual IP is `https://shmebd.mesproduction.com`.
- **K8S Dashboard Access**: The dashboard can be accessed for monitoring and managing the cluster.

Reports and Dashboard Cluster:

- **K8S Masters and Workers**:
  - **K8S Master 1, Master 2, and Master 3**: These nodes manage the orchestration and administration of worker nodes in the Reports and Dashboard Cluster.
  - **K8S Workers**: Multiple worker nodes (Worker 1, Worker 2, Worker 3, and possibly more) that handle the actual reporting and dashboard workloads.

- **HA Proxy**: Provides load balancing and high availability for the K8S masters in this cluster. It is accessed through port 6443.
- **Virtual IP 2**: This serves as the access point for MES report data. It can be accessed through **Port 443** for HTTPS traffic. The DNS entry for this virtual IP is `https://shmebd.mesreports.com`.
- **K8S Dashboard Access**: The dashboard can be accessed for monitoring and managing the cluster.

Access Points:

- **MES UI Access and External Integration**: Handled through Virtual IP 1 for the Production MES Cluster, which allows integration with external systems.
- **MES Reports Access**: Managed through Virtual IP 2 for the Reports and Dashboard Cluster, providing access to reports and dashboards.
- **Internet Access**: For external connectivity and access to the MES reports.

Cloud Integration:

The Reports and Dashboard Cluster also has access to the cloud, ensuring that report data can be stored and retrieved from cloud storage as needed.

Backup:

To export a snapshot or backup all the settings, refer Exporting a Snapshot of Settings.

Software Requirements for Hybrid Infrastructure

- **Kubernetes (K8S) Setup:**
  - **Kubernetes version:** Latest stable version.
  - **K8S Master Nodes:**
    - kube-apiserver
    - kube-controller-manager
    - kube-scheduler
    - etcd (for cluster state storage)
  - **K8S Worker Nodes:**
    - kubelet
    - kube-proxy
    - Container runtime (Docker, containerd, etc.)
  - **Networking:**
    - Kubernetes CNI (Container Network Interface) plugin (e.g., Calico, Flannel)
- **High Availability Proxy (HA Proxy):**
  - HAProxy version: Latest stable version.
  - Configuration for load balancing and failover.
- **Kafka (Data Flow):**
  - Kafka version: Latest stable version.
  - Zookeeper for Kafka coordination.
- **Ingress Controller:**
  - NGINX or Traefik for managing external access to services.
- **DNS:**
  - DNS entries for Virtual IP 1 and Virtual IP 2.
    - Virtual IP 1: https://k8smed.mesproduction.com
    - Virtual IP 2: https://k8smed.mesreports.com

- **Security:**
  - SSL/TLS certificates for secure communication over HTTPS (Port 443).
  - Firewalls to control access to the network.
- **Monitoring and Logging:**
  - Prometheus and Grafana for monitoring.
  - ELK stack (Elasticsearch, Logstash, Kibana) or EFK stack (Elasticsearch, Fluentd, Kibana) for logging.
- **Additional Tools:**
  - Helm for Kubernetes package management.
  - Kubectl for command-line management of Kubernetes clusters.

Note:

- Ensure redundancy and failover mechanisms are in place for critical components.
- Adjust the hardware configurations based on the specific workload and performance requirements.
- Regularly update and patch all software components to maintain security and stability.

Prerequisites

SmartHive Prerequisites for Application Deployment

Before deploying applications, it's crucial to have a functional Kubernetes cluster. Kubernetes is a powerful container orchestration tool that manages the deployment, scaling, and operation of application containers. To ensure a smooth deployment process, verify that the Kubernetes cluster is up and running, and all nodes are functioning correctly. Also, ensure that you have the necessary permissions and access rights to deploy applications to the cluster.

Before deploying applications on the SmartHive platform, ensure that your environment meets the following prerequisites:

Functional Kubernetes Cluster

Ensure that you have a functional Kubernetes cluster set up. Kubernetes is a powerful tool for managing application containers, handling deployment, scaling, and operation.

Required Packages

Install the following packages on each Ubuntu 22.04.03 VM:

- SmartHive Infrastructure Package
- SmartHive Application Installation Package

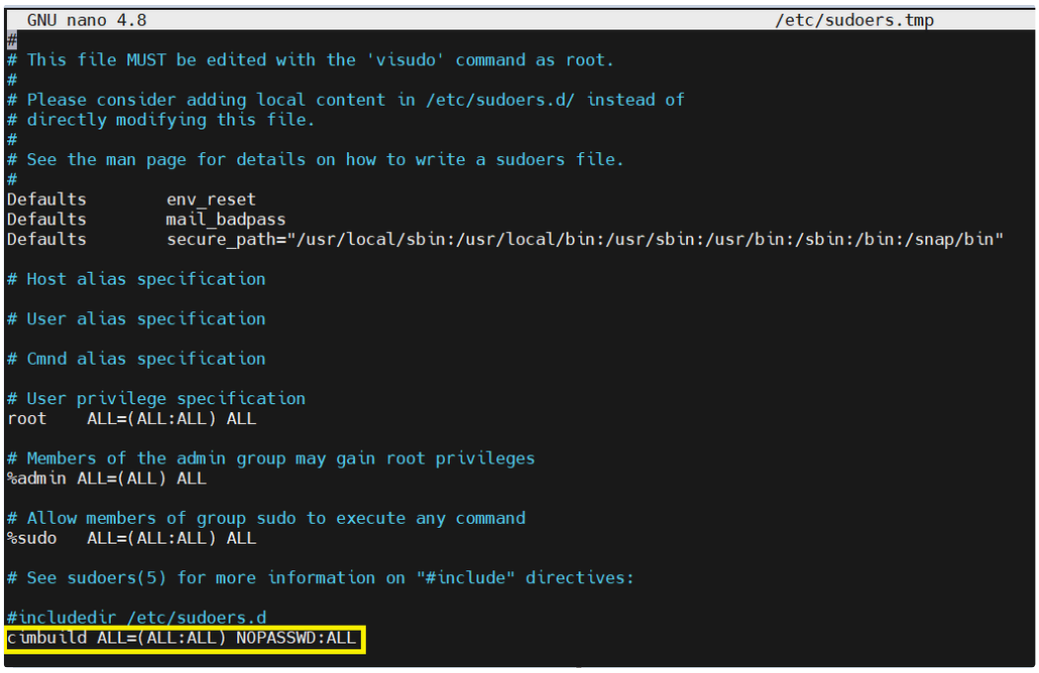
These packages are essential for the proper functioning of the SmartHive platform and must be installed before deployment.

Password-less sudo Setup

To make it easier to perform administrative tasks, you can set up password-less sudo access on your Ubuntu 22.04.03 VM. This allows authorized users to execute commands with superuser privileges without entering a password.

Steps to set up password-less sudo access:

1. The sudoers file is a configuration file that determines who can run sudo commands. You can open this file using a command called visudo. The sudoers file is located at /etc/sudoers.
   sudo visudo
2. Once the sudoers file is open, add a line at the end of the file to grant password-less sudo access to a specific user.
3. Replace **<username>** with the actual username of the user you want to grant access to.
4. The syntax for this line is **<username> ALL=(ALL:ALL**) NOPASSWD:ALL. For example, if the username is john, the line would look like this:
   john ALL=(ALL:ALL) NOPASSWD:ALL
5. Save the changes to the sudoers file and exit the editor.



SSH Setup for NCL Execution

To ensure secure communication between nodes and enable remote access for executing NCL (Node Configuration Language) scripts, you need to set up SSH (Secure Shell) on each Ubuntu 22.04.03 VM.

Following are the step-by-step instructions:

1. If the OpenSSH server package is not already installed, you can install it using the following command:
   sudo apt update
   sudo apt install openssh-server
2. Edit the SSH configuration file /etc/ssh/sshd_config using a text editor such as Nano:
   sudo nano /etc/ssh/sshd_config

Ensure the following settings are configured correctly in the sshd_config file:

- **PermitRootLogin no:** Disable root login for security reasons.
- **PasswordAuthentication no:** Disable password authentication for increased security (optional but recommended).
- **PubkeyAuthentication yes:** Enable public key authentication.

3. Save the file and exit the text editor.
4. Restart the SSH service to apply the changes:
   sudo systemctl restart ssh
5. Generate SSH keys if they have not been generated already. Use the following command to generate a new SSH key pair:
   ssh-keygen -b 4096
6. Copy the public key to the ~/.ssh/authorized_keys file on each VM to allow password-less SSH access. You can do this manually or using the ssh-copy-id command. For example:
   ssh-copy-id <remote_username>@<server_ip_address>
7. Replace **<remote_username>** with your remote server's username and **<server_ip_address>** with the IP address of the remote server.

Verification

To verify passwordless SSH connection with remote servers, use the following command:

ssh <username>@<remote_server_ip_address>

Enabling the Ubuntu VM Utility

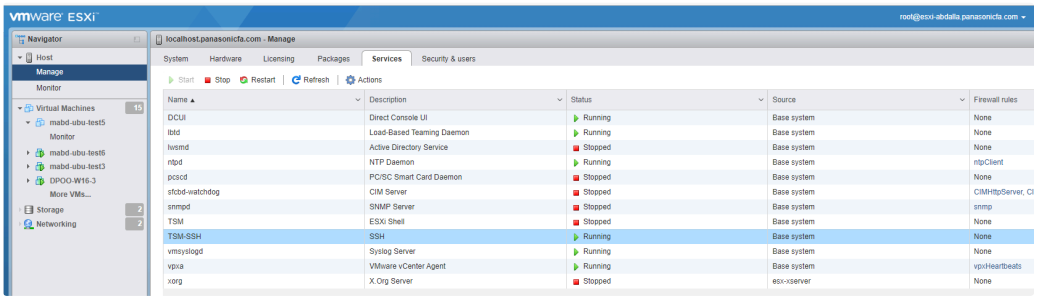
Deploying the SH High Availability Infrastructure requires a minimum of four virtual machines (VMs). Refer to the Hardware Configuration for details.

Before running the Ubuntu VM Utility on your VMware ESXi, complete the following prerequisite steps on the ESXi where the VM will be created.

Enable SSH on ESXi

Enable SSH on ESXi:

1. Navigate to ESXi UI page and go to "Manage" > "Services" tab.
2. Locate the entry called "TSM-SSH" and enable it.
3. Optionally, enable it to start up with the host by default inside the "Actions" dropdown (nested inside "Policy").



Prerequisite Steps

1. Enable "Guest IP Hack"

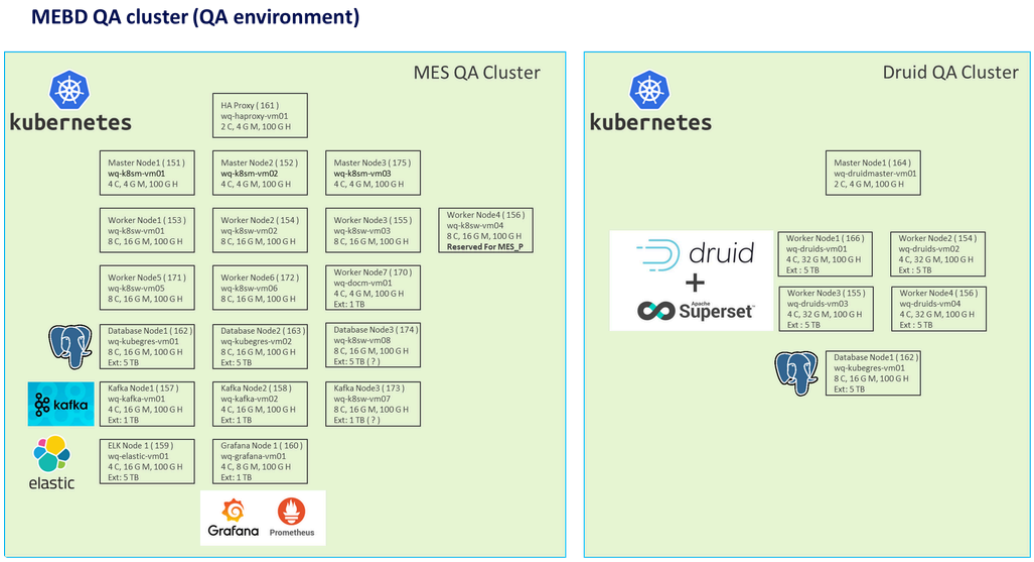
- Connect/login to your ESXi using SSH or PuTTY and run the following shell command:

1. Ensure You Can Edit the Firewall Configuration
2. Restore the Permissions and Reload the Firewall
3. Run the Attached Script

- Transfer the attached script to the ESXi server using SSH, PuTTY, or WinSCP and run it to address the above prerequisites.

H/W Configuration

This page provides a detailed breakdown of the hardware configuration for the MEBD QA environment, building upon the information presented in the table.



Infrastructure Summary

- **Environment:** QA (Quality Assurance)
- **Location:** Data Center (Factory)
- **Virtualization Platform:** VM Infra (assumed to be a virtual machine platform like VMware or KVM)

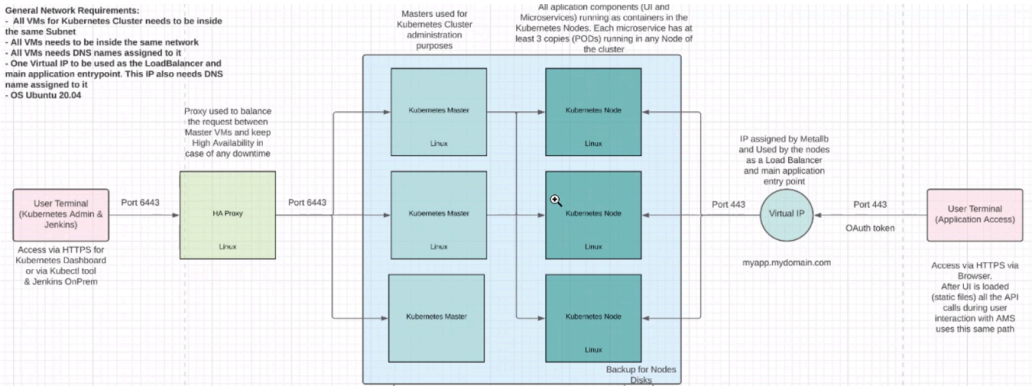
Components

The MEBD QA environment consists of several key components:

- **Kubernetes Cluster (**MES QA**):** This cluster manages containerized applications using Kubernetes orchestration.
  - **Masters:** Three VMs (wq-k8sm-vm01, wq-k8sm-vm02, wq-k8sm-vm03) handle control plane functions for the Kubernetes cluster.
  - **Workers:**
    - Four general-purpose VMs (wq-k8sw-vm01, wq-k8sw-vm02, wq-k8sw-vm03, wq-k8sw-vm04) - their specific purpose needs further investigation.
    - One VM (wq-k8sw-vm05) dedicated to workloads for PCONA (possibly a specific MEBD application).
    - Two unused VMs (wq-k8sw-vm06, wq-k8sw-vm07).
- **Kafka:** Two VMs (wq-kafka-vm01, wq-kafka-vm02) provide a distributed streaming platform for real-time data pipelines. One additional VM (wq-k8sw-vm07) is listed but unused.
- **Document OpenEBS (wq-docm-vm01):** This VM likely provides persistent storage for containerized applications using OpenEBS, potentially supporting PCONA functionality.
- **Elastic Search (wq-elastic-vm01):** This VM offers a search and analytics engine for log data or other relevant information.
- **Grafana (wq-grafana-vm01):** This VM provides a visualization platform for data exploration and monitoring.
- **HAProxy + Control Box (wq-haproxy-vm01):** This VM likely functions as a load balancer and potentially includes a control plane for managing the environment.

- **Druid Cluster (**Druid QA**):** This cluster is likely used for time-series data analysis.
  - **Master:** One VM (wq-druidmaster-vm01) coordinates the cluster operations.
  - **Data Nodes:** Four VMs (wq-druids-vm01, wq-druids-vm02, wq-druids-vm03, wq-druids-vm04) store and process the time-series data.
- **Data Stack:** This section might be for database services.
  - **PostgreSQL:** Two VMs (wq-kubegres-vm01, wq-kubegres-vm02) provide a relational database management system. One additional VM (wq-k8sw-vm08) is listed as unused.
  - **NTP Server (wq-ntpd-vm01 - Not currently configured):** This VM would synchronize time across the environment (currently inactive).

Solution Architecture Diagram for high availability



Components

- **User Terminals:** These represent workstations used to access and manage the Kubernetes environment.
- **Dubneten Admin Access (HTTPS):** This provides a secure web interface for managing the Kubernetes cluster.
- **Kubernetes Dashboard (HTTPS):** This is another web interface for monitoring and interacting with the Kubernetes cluster.
- **HAProxy + Load Balancer:** This acts as a single entry point for application traffic, distributing requests across multiple Kubernetes worker nodes.

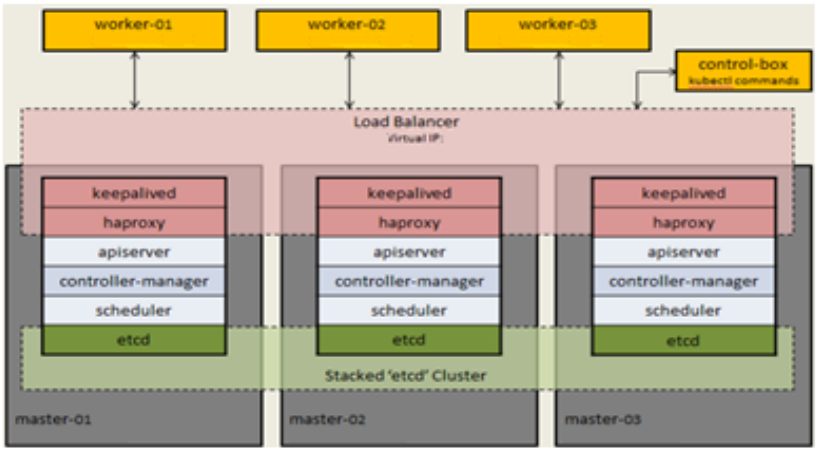
High Availability Considerations

- **Master High Availability:** The diagram only shows a single Kubernetes master node. For HA, it's recommended to have multiple masters in an active/passive or active/active configuration. This ensures that if a master node fails, another can take over seamlessly.
- **Worker Node High Availability:** The diagram doesn't show Kubernetes worker nodes. For HA, you'd typically have multiple worker nodes running pods (containers) for your applications. Kubernetes can automatically reschedule pods to healthy nodes if a worker node fails.
- **Disaster Recovery:** The diagram focuses on high availability within a single data center. For disaster recovery, you might consider replicating your Kubernetes cluster across geographically separate locations.
- **Network Redundancy:** The diagram doesn't show network components. A highly available network design with redundant paths and failover mechanisms is crucial for HA.
- **Storage Redundancy:** The diagram doesn't show storage. For persistent data storage used by applications in pods, you'd typically implement a redundant storage solution.

High Availability Infrastructure

High Availability (HA) infrastructure is a set of technologies and configurations designed to keep a system operational for a very high percentage of time. It aims to minimize downtime caused by hardware or software failures. Following are the key aspects of HA infrastructure:

- **Redundancy:** This refers to having multiple components that can perform the same function. In case of a failure in the primary component, a redundant component takes over to ensure uninterrupted service. Redundancy can be applied to various components like servers, storage, network connections, and even software applications.
- **Fault Tolerance:** This refers to a system's ability to continue operating even when one or more of its components malfunction. Fault tolerance mechanisms can detect failures, isolate them, and potentially recover from them without human intervention.
- **Scalability:** HA infrastructure should be designed to grow and adapt to changing needs. This might involve adding more redundant components to handle increased load or distribute workloads more efficiently.



The diagram explains a stacked etcd cluster, which is a key component for achieving high availability in a Kubernetes environment. Let's break down the functionality of each element in the image:

- **etcd Clusters:** Etcd is a distributed key-value store that acts as the central storage for Kubernetes cluster data. It stores critical information about the cluster configuration, including pod deployments, service definitions, and secrets. A stacked etcd cluster refers to running multiple etcd instances working together. This redundancy ensures that even if one etcd node fails, the remaining nodes can continue operating and serving requests.
- **Control Plane:** The control plane manages the Kubernetes cluster and consists of several components, including the API server, scheduler, and etcd. The image depicts three Kubernetes masters (master-01, master-02, master-03), likely representing an etcd cluster in an active/active configuration. This means all three etcd nodes can process requests concurrently, improving availability and scalability.
- **Worker Nodes:** The worker nodes are responsible for running containerized applications in pods. These are not shown in the image, but they are crucial components in a Kubernetes cluster.

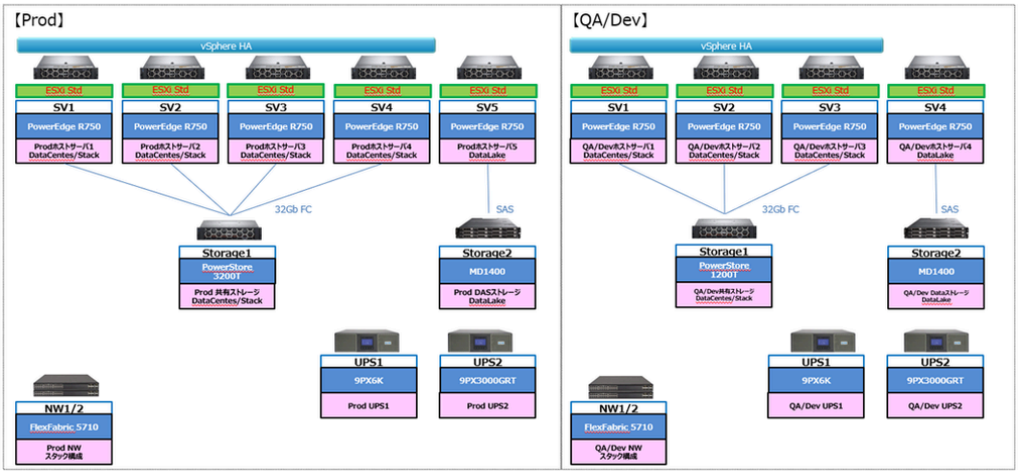
Benefits of using a stacked etcd cluster in a Kubernetes environment

- **Improved Availability:** If a single etcd node fails, the cluster remains operational because the other nodes can continue serving requests.
- **Scalability:** You can easily add more etcd nodes to the cluster to handle increased load or improve read/write performance.
- **Data Consistency:** Etcd ensures strong consistency by replicating data across all nodes in the cluster. This guarantees that all nodes always have the same up-to-date information.

System Map

The systems map illustrates the configuration of hardware components, storage solutions, network infrastructure, and power supply units for both production and QA/Development environments. Both environments utilize similar Dell PowerEdge servers and FlexFabric switches, with differences primarily in the specific models of PowerStore storage arrays. The map ensures redundancy and high availability through vSphere HA and dual UPS units.

The following is the system map with connected devices:



Production Environment (Prod)

- **Servers (ESXi Std)**:
  - **SV1, SV2, SV3, SV4**: Dell PowerEdge R750 servers connected to shared storage via vSphere HA (High Availability).
  - **SV5:** Dell PowerEdge R750 server connected to the Direct-Attached Storage (DAS).
- **Storage**:
  - **Storage1 (PowerStore 3200T)**: Shared storage for data centers and stacks, connected via 32GB FC (Fiber Channel).
  - **Storage2 (MD1400)**: DAS for DataLake, connected via SAS (Serial Attached SCSI).
- **Network**:
  - **NW1/2 (FlexFabric 5710)**: Network switches for stack connectivity.
- **Power Supply**:
  - **UPS1 (9PX6K)**: Uninterruptible power supply for redundancy.
  - **UPS2 (9PX3000GRT)**: Additional UPS for redundancy.

QA/Development Environment (QA/Dev)

- **Servers (ESXi Std)**:
  - **SV1, SV2, SV3**: Dell PowerEdge R750 servers connected to shared storage via vSphere HA.
  - **SV4:** Dell PowerEdge R750 server connected to the DAS.
- **Storage**:
  - **Storage1 (PowerStore 1200T)**: Shared storage for data centers and stacks, connected via 32GB FC.
  - **Storage2 (MD1400)**: DAS for DataLake, connected via SAS.
- **Network**:

    - **NW1/2 (FlexFabric 5710)**: Network switches for stack connectivity.
- **Power Supply**:
    - **UPS1 (9PX6K)**: Uninterruptible power supply for redundancy.
    - **UPS2 (9PX3000GRT)**: Additional UPS for redundancy.

Key Connections

- **32GB FC**: Indicates high-speed fiber channel connections for shared storage.
- **SAS**: Serial Attached SCSI connections for direct-attached storage.
- **vSphere HA**: Virtualization platform providing high availability for connected servers.

Installation

This section covers various step-by-step installation instructions for different system components.

Infrastructure Installation

Retrieve Deployment Tar File

- To begin the deployment process, you need to obtain the deployment tar file from the DevOps team. This file is essential as it contains all the necessary infrastructure components required for setting up the SmartHive platform.
- Once you have received the deployment tar file, you can extract its contents to reveal the "multi-server" folder. This folder is pivotal as it houses all the components needed for the infrastructure deployment.
- Inside the "multi-server" folder, you will find various files and directories that are crucial for the deployment process. These components are organized and packaged to streamline the installation process and ensure that all necessary files are included.
- Extracting the deployment tar file and locating the "multi-server" folder is the first step towards setting up the SmartHive platform. This folder serves as the foundation for the infrastructure deployment and contains everything needed to proceed with the installation process.

Accessing Kubernetes Scripts and Docker Images

By accessing the Kubernetes scripts and Docker images provided in the "multi-server" folder, you can effectively deploy and manage your Kubernetes infrastructure and applications with ease.

Kubernetes Scripts

To access the Kubernetes scripts required for deployment, navigate to the "multi-server" folder. These scripts are located in the "playbooks" folder and are structured as Ansible playbooks. Ansible playbooks provide a declarative configuration management approach, making it easier to automate the deployment and configuration of Kubernetes components across your infrastructure.

Following are the steps to navigate to the "multi-server" folder:

1. Copy/transfer the HA Installation package to the Control Box:

- Please ftp or scp the multi-server.tar.gz to the Control Box VM.

2. Untar the installation package tar file:

- Untar the multi-server.tar.gz file using the following command:

tar -xf multi-server.tar.gz -C multi-server

3. Please refer to the screenshot below for the contents of the multi-server package once it is untarred.

Docker Images and Tar Files

In addition to the Kubernetes scripts, you'll also find Docker images and tar files for services like Helm in the "Downloads" folder. Docker images are used to create containers, which are lightweight, portable, and isolated environments that encapsulate an application and its dependencies. Helm is a package manager for Kubernetes that simplifies the deployment and management of applications on Kubernetes clusters. The tar files in the "Downloads" folder contain the necessary files and configurations needed for these services to run within the Kubernetes environment.

Usage

To use these scripts and files, ensure you have the necessary permissions and access rights. You may need to modify the scripts or configurations to suit your specific deployment environment. Additionally, refer to the documentation provided with these resources for detailed instructions on their usage and configuration.

Security Considerations

When working with Docker images and Kubernetes scripts, it's essential to follow best practices for security. This includes regularly updating your images and scripts, using secure communication protocols, and restricting access to sensitive information and resources.

Install Kubernetes Framework

The start script automates the installation of the Kubernetes framework using ansible. This process ensures that all necessary configurations are applied to both master and worker nodes. The framework includes components such as kubelet, kubectl, and kubeadm, which are essential for managing the Kubernetes cluster.

Initializing the Kubernetes Cluster

Once the framework is installed, the next step is to initialize the Kubernetes cluster using the kubeadm init command. This command sets up the initial cluster state and generates the kubeconfig file (admin.conf) in the /etc/kubernetes/ folder. The kubeconfig file contains authentication details and cluster information needed to access the cluster.

Validating the Cluster

To ensure that the Kubernetes cluster is successfully initialized and all nodes are in the "Ready" state, execute the command sudo kubectl get nodes. This command provides an overview of the cluster nodes and their status. All nodes should be in the "Ready" state before proceeding with further configurations.

Installing Infrastructure Components

After the Kubernetes cluster is set up, the start script proceeds to install additional infrastructure components such as Kubegres and the K8S dashboard. These components enhance the functionality of the Kubernetes cluster and provide additional features for managing and monitoring applications.

Validating Installed Services

To verify that all installed services are running correctly, use the command sudo kubectl get pods -A. This command lists all pods in the cluster, including those for the installed infrastructure components. Ensure that all pods are in the "Running" state and that there are no errors.

Accessing the Dashboard

Optionally, you can refer to the installation guide for the Dashboard component to access the online dashboard. The dashboard provides a graphical user interface for managing and monitoring the Kubernetes cluster, making it easier to visualize cluster metrics and perform administrative tasks.

Step by step instructions for SmartHive Infrastructure Deployment

1. Obtain the deployment tar file from the DevOps team and extract it to find the "multi-server" folder. This folder contains all the infrastructure components to be deployed in tar format.

- Additionally, Kubernetes scripts can be found in the "playbooks" folder as Ansible playbooks. Docker images and tar files for services like Helm are located in the "Downloads" folder.

- The "functions.sh" script contains all the actions and commands needed to set up the infrastructure. It is initialized by running "start_if_installer.sh".

2. To begin the installation of the SmartHive infrastructure, execute the start_if_installer.sh script as follows:

```
cimbuild@vmcb:~/multi-server$ ./start_if_installer.sh

[INFO]: Reading User Inputs ...

[INFO]: Preparing Control Box ...

[INFO]: Installing Pre-Requisites ...
  --> Docker already installed!
  --> Docker compose already installed!
  --> Psql already installed!
  --> Kubectl already installed!
  --> Python3 already installed!
  --> Setfacl already installed!
  --> Sshpass already installed!
  --> netaddr already installed!
  --> Pip already installed!
  --> Ansible already installed!
  --> Configuring docker registry
  --> Restarting docker service

[INFO]: Validating Pre-Requisites ...

[INFO]: Building APT Repo on Control Box ...
  --> Copying apt packages
  --> Loading docker image
  --> Docker container already running. Restarting...
Restarting panasonic-apt-repo ... done
  --> Updating apt repo

[INFO]: Building Docker Registry on Control Box ...
  --> Loading docker image
  --> Docker container already running. Restarting...
Restarting panasonic-registry ... done

[INFO]: Loading Docker Images to Docker Registry on Control Box ...
WARNING! Your password will be stored unencrypted in /home/cimbuild/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
  --> Loading kubernetes images
Loaded image: flannelcni/flannel:v0.20.2
Loaded image: flannelcni/flannel-cni-plugin:v1.2.0
Loaded image: registry.k8s.io/coredns/coredns:v1.9.3
Loaded image: registry.k8s.io/cpa/cluster-proportional-autoscaler-amd64:1.8.5
Loaded image: registry.k8s.io/dns/k8s-dns-node-cache:1.22.18
Loaded image: registry.k8s.io/kube-apiserver:v1.24.1
Loaded image: registry.k8s.io/kube-controller-manager:v1.24.1
Loaded image: registry.k8s.io/kube-proxy:v1.24.1
```

3. After installing the prerequisites and verifying the required services like Docker, the start script will initiate the installation of the Kubernetes framework using Ansible across all the necessary master and worker nodes:

```
TASK [container-engine/docker : ensure service is started if docker packages are already present] ********
ok: [node5]
ok: [node4]
ok: [node6]
Monday 01 April 2024  12:32:15 +0000 (0:00:01.518)       0:05:09.854 **********
Monday 01 April 2024  12:32:15 +0000 (0:00:00.000)       0:05:09.855 **********
Monday 01 April 2024  12:32:15 +0000 (0:00:00.221)       0:05:10.077 **********
Monday 01 April 2024  12:32:16 +0000 (0:00:00.483)       0:05:10.561 **********

TASK [container-engine/docker : Create docker service systemd directory if it doesn't exist] *************
changed: [node3]
ok: [node4]
ok: [node5]
changed: [node1]
ok: [node6]
changed: [node2]
Monday 01 April 2024  12:32:17 +0000 (0:00:01.368)       0:05:11.930 **********
Monday 01 April 2024  12:32:17 +0000 (0:00:00.487)       0:05:12.417 **********

TASK [container-engine/docker : get systemd version] ***************************************************
ok: [node3]
ok: [node4]
ok: [node5]
ok: [node6]
ok: [node1]
ok: [node2]
Monday 01 April 2024  12:32:19 +0000 (0:00:01.427)       0:05:13.845 **********

TASK [container-engine/docker : Write docker.service systemd file] *************************************
changed: [node3]
changed: [node1]
ok: [node5]
ok: [node4]
ok: [node6]
```

- This installation process will configure Kubernetes and install kubelet, kubectl, and kubeadm.

4. Once these packages are installed, the kubeadm init step is executed to start the Kubernetes cluster. This step also creates the kubeconfig file (admin.conf) under the /etc/kubernetes/ folder.

```
TASK [kubernetes/control-plane : Set fact first_kube_control_plane] ********************
ok: [node1]
ok: [node2]
Monday 01 April 2024  13:11:14 +0000 (0:00:00.196)       0:15:19.830 **********
Monday 01 April 2024  13:11:14 +0000 (0:00:00.159)       0:15:19.989 **********

TASK [kubernetes/control-plane : kubeadm | Check if kubeadm has already run] ***********
ok: [node1]
ok: [node2]
Monday 01 April 2024  13:11:15 +0000 (0:00:00.660)       0:15:20.650 **********
Monday 01 April 2024  13:11:15 +0000 (0:00:00.148)       0:15:20.798 **********
Monday 01 April 2024  13:11:15 +0000 (0:00:00.169)       0:15:20.968 **********

TASK [kubernetes/control-plane : kubeadm | aggregate all SANs] *************************
ok: [node1]
ok: [node2]
Monday 01 April 2024  13:11:16 +0000 (0:00:00.445)       0:15:21.413 **********
Monday 01 April 2024  13:11:16 +0000 (0:00:00.097)       0:15:21.510 **********
Monday 01 April 2024  13:11:16 +0000 (0:00:00.090)       0:15:21.601 **********
Monday 01 April 2024  13:11:16 +0000 (0:00:00.093)       0:15:21.694 **********

TASK [kubernetes/control-plane : set kubeadm_config_api_fqdn define] *******************
ok: [node1]
ok: [node2]
Monday 01 April 2024  13:11:16 +0000 (0:00:00.133)       0:15:21.827 **********

TASK [kubernetes/control-plane : Set kubeadm api version to v1beta3] *******************
ok: [node1]
ok: [node2]
Monday 01 April 2024  13:11:16 +0000 (0:00:00.107)       0:15:21.934 **********

TASK [kubernetes/control-plane : kubeadm | Create kubeadm config] **********************
changed: [node1]
changed: [node2]
Monday 01 April 2024  13:11:18 +0000 (0:00:01.639)       0:15:23.574 **********
Monday 01 April 2024  13:11:18 +0000 (0:00:00.099)       0:15:23.673 **********
Monday 01 April 2024  13:11:18 +0000 (0:00:00.118)       0:15:23.792 **********
Monday 01 April 2024  13:11:18 +0000 (0:00:00.113)       0:15:23.905 **********
Monday 01 April 2024  13:11:18 +0000 (0:00:00.111)       0:15:24.017 **********
Monday 01 April 2024  13:11:18 +0000 (0:00:00.132)       0:15:24.150 **********
Monday 01 April 2024  13:11:18 +0000 (0:00:00.120)       0:15:24.270 **********
Monday 01 April 2024  13:11:19 +0000 (0:00:00.124)       0:15:24.395 **********
Monday 01 April 2024  13:11:19 +0000 (0:00:00.117)       0:15:24.512 **********
```

- Upon successful execution of the Kubernetes scripts, the following screen should be displayed:

```
PLAY RECAP *********************************************************************************************************************************
localhost                  : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
node1                      : ok=642  changed=134  unreachable=0    failed=0    skipped=1106 rescued=0    ignored=5
node2                      : ok=406  changed=84   unreachable=0    failed=0    skipped=746  rescued=0    ignored=1
node3                      : ok=381  changed=81   unreachable=0    failed=0    skipped=663  rescued=0    ignored=1
node4                      : ok=365  changed=27   unreachable=0    failed=0    skipped=645  rescued=0    ignored=1
node5                      : ok=374  changed=26   unreachable=0    failed=0    skipped=645  rescued=0    ignored=1
node6                      : ok=374  changed=26   unreachable=0    failed=0    skipped=645  rescued=0    ignored=1

Wednesday 03 April 2024  17:41:04 +0000 (0:00:00.353)       0:18:11.584 *******
===============================================================================
container-engine/docker : ensure docker packages are installed ---------------------------------------------------- 28.78s
kubernetes/control-plane : kubeadm | Initialize first master ------------------------------------------------------ 18.75s
container-engine/cri-dockerd : cri-dockerd | restart docker.service ----------------------------------------------- 17.29s
kubernetes/kubeadm : Join to cluster ------------------------------------------------------------------------------ 15.37s
container-engine/validate-container-engine : Populate service facts ----------------------------------------------- 14.89s
Gather necessary facts (hardware) ---------------------------------------------------------------------------------- 12.27s
kubernetes/preinstall : Install packages requirements -------------------------------------------------------------- 9.60s
container-engine/cri-dockerd : extract_file | Unpacking archive ---------------------------------------------------- 8.28s
kubernetes/node : look up docker cgroup driver --------------------------------------------------------------------- 8.20s
container-engine/crictl : extract_file | Unpacking archive --------------------------------------------------------- 7.91s
network_plugin/cni : CNI | Copy cni plugins ------------------------------------------------------------------------ 7.81s
kubernetes-apps/metrics_server : Metrics Server | Create manifests ------------------------------------------------- 7.76s
kubernetes-apps/metrics_server : Metrics Server | Apply manifests -------------------------------------------------- 7.50s
download : download_container | Download image if required -------------------------------------------------------- 6.58s
kubernetes-apps/ansible : Kubernetes Apps | Lay Down CoreDNS templates --------------------------------------------- 6.43s
kubernetes/preinstall : Ensure kube-bench parameters are set ------------------------------------------------------- 6.33s
etcd : Configure | Ensure etcd is running -------------------------------------------------------------------------- 5.77s
etcd : Configure | Check if etcd cluster is healthy ---------------------------------------------------------------- 5.68s
kubernetes-apps/network_plugin/flannel : Flannel | Wait for flannel subnet.env file presence ---------------------- 5.67s
download : extract_file | Unpacking archive ------------------------------------------------------------------------ 5.50s
Waiting for the cluster...
```

- If the Kubernetes installation fails for any reason, the installation process can be terminated, issues can be troubleshooted, and the start script can be restarted.

5. After successfully installing Kubernetes cluster, we can validate if the nodes in cluster are up by the following command, and check the “Ready” state:

>> sudo kubectl get nodes

```
cimbuild@node1:~$ sudo kubectl get nodes
NAME     STATUS   ROLES           AGE   VERSION
node1    Ready    control-plane   22d   v1.24.1
node10   Ready    <none>          21d   v1.24.1
node11   Ready    <none>          21d   v1.24.1
node12   Ready    <none>          21d   v1.24.1
node13   Ready    <none>          22d   v1.24.1
node14   Ready    <none>          21d   v1.24.1
node15   Ready    <none>          22d   v1.24.1
node16   Ready    <none>          22d   v1.24.1
node17   Ready    <none>          22d   v1.24.1
node18   Ready    <none>          21d   v1.24.1
node19   Ready    <none>          22d   v1.24.1
node2    Ready    control-plane   22d   v1.24.1
node20   Ready    <none>          22d   v1.24.1
node21   Ready    <none>          21d   v1.24.1
node22   Ready    <none>          21d   v1.24.1
node23   Ready    <none>          22d   v1.24.1
node24   Ready    <none>          22d   v1.24.1
node25   Ready    <none>          21d   v1.24.1
node26   Ready    <none>          21d   v1.24.1
node27   Ready    <none>          22d   v1.24.1
node28   Ready    <none>          22d   v1.24.1
node3    Ready    control-plane   22d   v1.24.1
node4    Ready    <none>          22d   v1.24.1
node5    Ready    <none>          21d   v1.24.1
node6    Ready    <none>          22d   v1.24.1
node7    Ready    <none>          21d   v1.24.1
node8    Ready    <none>          22d   v1.24.1
node9    Ready    <none>          21d   v1.24.1
```

- After this, the installation process returns to the start_if_installer.sh script, which then extracts infrastructure zipped folders one by one and installs necessary infrastructure components such as Kubegres and K8S dashboard.

```
-------------------------------------
        INSTALLING THE OPENEBS
-------------------------------------
openebs/
openebs/k8s_openebs_install.sh
openebs/mayan-sample/
openebs/mayan-sample/1-simple-dynamic-provisioning/
openebs/mayan-sample/1-simple-dynamic-provisioning/1-namespace.yml
openebs/mayan-sample/1-simple-dynamic-provisioning/3-example-dynamic-pvc.yml
openebs/mayan-sample/1-simple-dynamic-provisioning/4-mayan.yml
openebs/mayan-sample/1-simple-dynamic-provisioning/2-openebs-hostpath-sc.yml
openebs/mayan-sample/3-dynamic-provisining-with-replicas-scalable/
openebs/mayan-sample/3-dynamic-provisining-with-replicas-scalable/6-nfs-podsecuritypolicy-rbac.yml
openebs/mayan-sample/3-dynamic-provisining-with-replicas-scalable/8-nfs-storageclass.yml
openebs/mayan-sample/3-dynamic-provisining-with-replicas-scalable/4-jiva-storageclass.yml
openebs/mayan-sample/3-dynamic-provisining-with-replicas-scalable/1-namespace.yml
openebs/mayan-sample/3-dynamic-provisining-with-replicas-scalable/10-mayan.yml
openebs/mayan-sample/3-dynamic-provisining-with-replicas-scalable/5-example-dynamic-ha-pvc.yml
openebs/mayan-sample/3-dynamic-provisining-with-replicas-scalable/9-nfs-pvc.yml
openebs/mayan-sample/3-dynamic-provisining-with-replicas-scalable/3-jiva-volumepolicy.yml
openebs/mayan-sample/3-dynamic-provisining-with-replicas-scalable/7-nfs-provisioner.yml
openebs/mayan-sample/3-dynamic-provisining-with-replicas-scalable/2-openebs-hostpath-sc.yml
openebs/mayan-sample/2-dynamic-provisining-with-replicas/
openebs/mayan-sample/2-dynamic-provisining-with-replicas/6-mayan.yml
openebs/mayan-sample/2-dynamic-provisining-with-replicas/4-jiva-storageclass.yml
openebs/mayan-sample/2-dynamic-provisining-with-replicas/1-namespace.yml
openebs/mayan-sample/2-dynamic-provisining-with-replicas/5-example-dynamic-ha-pvc.yml
openebs/mayan-sample/2-dynamic-provisining-with-replicas/3-jiva-volumepolicy.yml
openebs/mayan-sample/2-dynamic-provisining-with-replicas/2-openebs-hostpath-sc.yml
openebs/enable_iscsi.sh
openebs/m-exporter_3.4.0.tar
openebs/linux-utils_v3.4.0.tar
openebs/readme.md
```

```
-------------------------------------
        INSTALLING THE DASHBOARD
-------------------------------------
dashboard/
dashboard/k8s_dashboard_install.sh
dashboard/dashboard.yaml
dashboard/dashboard-adminuser.yaml
dashboard/images/
dashboard/images/metrics-scraper.v1.0.8.tar.gz
dashboard/images/dashboard.v2.7.0.tar.gz
dashboard/ingress-deploy.yml
dashboard/recommended.yaml
------------------------- Start script - Create variables ------------------------------
Container Registry:  10.140.79.131
Home path:  /home/cimbuild/multi-server/dashboard
------------------------- Prepare the VMs connection -----------------------------------
------------------------- Deploy Ingress yml -------------------------------
namespace/ingress-nginx created
serviceaccount/ingress-nginx created
serviceaccount/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
configmap/ingress-nginx-controller created
service/ingress-nginx-controller created
service/ingress-nginx-controller-admission created
deployment.apps/ingress-nginx-controller created
job.batch/ingress-nginx-admission-create created
```

6. Once all these components are deployed, validate the pods for all the installed services using the following command:

>> sudo kubectl get pods -A

```
cimbuild@node1:~$ sudo kubectl get pods -A
NAMESPACE              NAME                                                READY   STATUS             RESTARTS
        AGE
elastic-system         elastic-operator-0                                  1/1     Running            9 (8h ago)
        16h
if                     panasonic-if-adapter-registry-api-5486c987fd-wj7k7  1/1     Running            0
        8h
if                     panasonic-if-manager-api-6dbf46b86b-8669t           1/1     Running            0
        19d
if                     panasonic-if-manager-monitoring-api-774b8b784b-xnjjk  1/1     Running          0
        19d
if                     panasonic-if-manager-ui-6698769584-gbdv2            1/1     Running            0
        8h
ingress-nginx          ingress-nginx-controller-877dfd8cc-l77r2            1/1     Running            1 (13d ago
)       13d
istio-system           istio-ingressgateway-685fcc78c-9zhqk                1/1     Running            0
        8h
istio-system           istio-operator-66698fb64f-v8d5m                     1/1     Running            54 (9h ago
)       21d
istio-system           istiod-75f586d698-vjkf6                             1/1     Running            0
```

- Alternatively, refer to the installation of the Dashboard component, which provides details on accessing the online dashboard.

```
------------------------- Get k8s cluster config file -------------------------------
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMvakNDQWVhZ0F3SUJBZ0lCQURBTkJna3Foa2lHOXcwQkFRc0ZBRE
    server: https://10.140.79.149:6443
  name: cluster.local
contexts:
- context:
    cluster: cluster.local
    user: kubernetes-admin
  name: kubernetes-admin@cluster.local
current-context: kubernetes-admin@cluster.local
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURJVENDQWdtZ0F3SUJBZ0lJSHNleHFzdzBCRTR3RFFZSktvWklodmNOQ
    client-key-data: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcEFJQkFBS0NBUUVBNTdTbFZJZ0dDalZuVUx0aFlCdU9kaDJhZ2RHOVpGW
------------------------- Create access token --------------------------------
eyJhbGciOiJSUzI1NiIsImtpZCI6IkVQeTY5M1czelRQbXVHUmxuSkpLNWdVWDZxZHFuZ05FSzhVWE14Tl9pd2MifQ.eyJhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGV
From the local machine, after configure the kubectl to connect to the cluster run:
'sudo kubectl proxy'
Use a browser and the token to access the dashboard with the url:
http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/
```

Core / Platform Applications Installation

SmartHive Core App / Support App / Adapter Fresh / Upgrade Deployment

Obtain Deployment Scripts

Access the deployment scripts from Bitbucket at the provided URL. These scripts are essential for deploying the applications on the Customer Control Box. The deployment scripts include the necessary configurations and commands to automate the deployment process, ensuring consistency and reliability.

Login to Linux Machine

Log in to the Linux machine that has access to the Customer Control Box. Ensure that you have the necessary permissions and credentials to access the machine and perform deployment tasks.

Retrieve Code

Retrieving code on the Customer Control Box consists of three files:

Apps

The Apps file is used to list all the applications to be deployed or updated. Each entry should follow the format **<app name>-<version/tag>**. The file serves as a list of applications to be deployed or updated.

Config.ini

The Config.ini file contains customer-specific properties used by the deployment script. Following is the description of each property:

- **CB_HOSTNAME:** Control Box Hostname
- **CB_USERNAME:** Username to login to the Control Box
- **CB_PASSWORD:** Password to login to the Control Box
- **K8S_MASTER:** IP address of the Kubernetes Master VMs. If there are more than 1 Master, separate the IP addresses by space.
- **K8S_WORKER:** IP address of the Kubernetes Workers. If there are more than 1 workers, separate the IP addresses by space (same as the K8S_Master).
- **K8S_VIP:** Virtual IP Address. Provided by IT.
- **K8S_VHOSTNAME:** Fully Qualified Domain Name (FQDN) of the Virtual Host (VIP). Provided by IT.
- **HA_PROXY_IP:** IP address of the HA Proxy VM
- **DASHBOARD_SERVER:** Same IP address as the first K8S_WORKER
- **KUBEGRES_SERVER:** IP address of the kubegres server. If there are more than 1, separate the IP addresses by space. These are also K8S worker IP addresses.
- **KUBEKAFKA_SERVER:** IP address of the kubekafka server. If there are more than 1, separate the IP addresses by space. These are also K8S worker IP addresses.
- **ELK_SERVER:** IP address of the ELK server. It will be one of the K8S worker IP addresses.
- **KEYCLOAK_SERVER:** Same IP address as the first K8S_WORKER
- **OPENEBS_SERVER:** Same IP address as the first K8S_WORKER
- **REDIS_SERVER:** Same IP address as the first K8S_WORKER
- **ISTIO_SERVER:** Same IP address as the first K8S_WORKER

Deploy.sh

The Deploy.sh is used to execute the deployment script. The deploy.sh script reads data from apps and config.ini to deploy the apps. The syntax for executing the deployment script should be included wherever applicable. For debugging purposes, add the following instructions:

- To enable debugging, set the DEBUG variable to true in the deploy.sh script.
- Run the script with the -x flag to print each command before executing it, which can help trace and debug issues.

Run Deployment Script

Execute the deployment script using ./deploy.sh and follow the instructions provided. The script will start the deployment process, which includes pulling the latest code from the repository, building the applications, and deploying them to the target environment. Ensure that the script completes successfully without any errors.

Validate Deployment

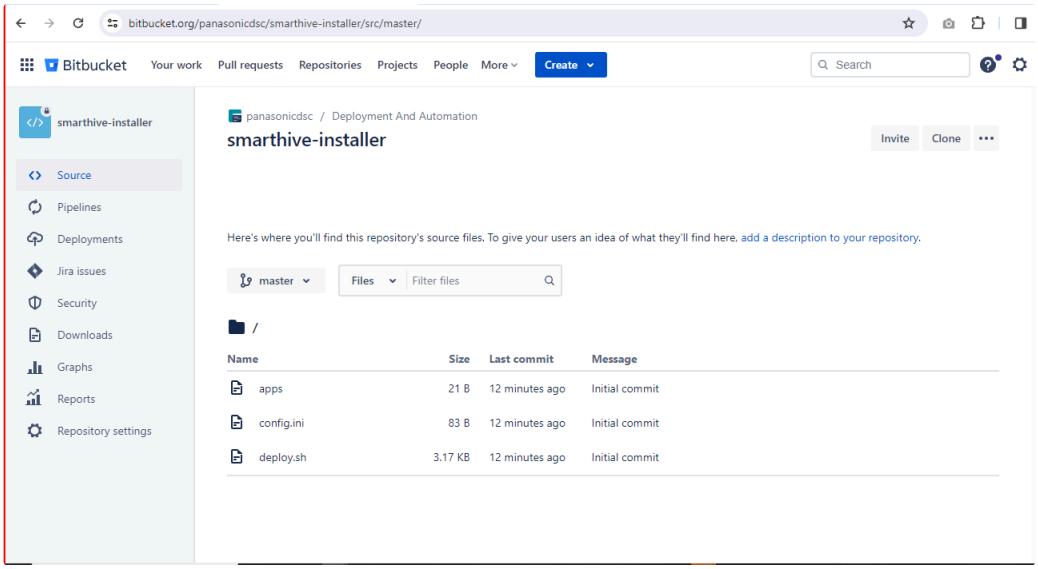
After deployment, validate the deployment status/logs from Jenkins by opening a browser and accessing the URL: http://<CB_HOSTNAME>:8080/job/panasonic.qualitycenter.solutionapp.ui-pipeline. This step ensures that the deployment was successful and that the applications are running correctly in the target environment.

Check Application

Check the application on the main/ui page at https://<K8S_VHOSTNAME>/main/ui. Login with the credentials specified in config.ini (UI_USER/UI_PWD). Once logged in, give required permissions to the newly added app (not required for upgrades) by navigating to user management, selecting a user with admin access, clicking on 'Privileges', selecting Application, Features, and Access level, and then clicking on SAVE button. Finally, refresh the main/ui page to see the new app available on the main/ui page.

Deployment of SmartHive Core/Platform Application

1. Obtain the deployment scripts from Bitbucket at the following URL:
   https://bitbucket.org/panasonicdsc/smarthive-installer/src/master/



2. Login to the Linux machine that has access to the Customer Control Box.
3. Retrieve the code on the Customer Control Box. The code comprises the following three files:
   a. **Apps**: Update this file with all the applications to be deployed, following the format **<app name>-<version/tag>**. For example, panasonic-qualitycenter-solutionapp-ui-0.0.54, where panasonic-qualitycenter-solutionapp-ui is the app name and 0.0.54 is the version/tag.
   b. **Config.ini**: This file contains customer-specific properties, such as:

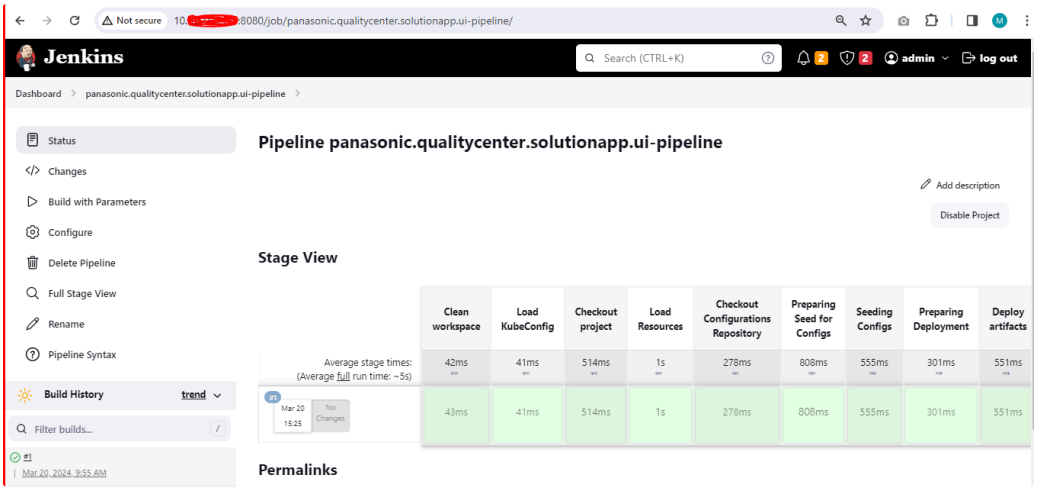
- CB_HOSTNAME
- CB_USERNAME
- CB_PASSWORD
- K8S_MASTER
- K8S_VHOSTNAME
- UI_USER
- UI_PWD

c. **Deploy.sh:** This script reads data from **apps** and **config.ini** to deploy the apps on the target environment.

i. Run the script using **./deploy.sh** and follow the instructions.

```
cimbuild@infra-ubu-test5:~/app_deployment/customer$ ./deploy.sh
[INFO]: Deploying
   APP: panasonic-qualitycenter-solutionapp-ui
   TAG: 0.0.54
[CHK ]: Press [Y/y] to proceed: y
proceeding ...
[INFO]: Copying panasonic-qualitycenter-solutionapp-ui-0.0.54.tar.gz to <CB_HOSTNAME>
panasonic-qualitycenter-solutionapp-ui-0.0.54.tar.gz 100% 18MB 1.6MB/s 00:11
[INFO]: Extracting Offline Package on <CB_HOSTNAME>
[INFO]: Updating Config.ini
config.ini 100% 188 1.0KB/s 00:00
[INFO]: Pods Before Upgrade
panasonic-qualitycenter-solutionapp-ui-864fd86698-phvzn 1/1 Running 0 48d 0.0.40
[INFO]: Starting Deployment
[INFO]: Login to docker registry...
Login Succeeded
[INFO]: Pushing panasonic-qualitycenter-solutionapp-ui:0.0.54 to docker registry...
[INFO]: Jenkins running at http://<CB_HOSTNAME>:8080/
[INFO]: Triggering Pipeline mwsinv-ingress-single-seed
Build Status: SUCCESS
[INFO]: Triggering Pipeline mwsinv-istio-single-seed
Build Status: SUCCESS
[INFO]: Triggering Pipeline panasonic.qualitycenter.solutionapp.ui-pipeline with Tag
Build Status: SUCCESS
[INFO]: Running App Permissions
[INFO]: Running App Settings
[INFO]: Running App NotificationTrigger
[INFO]: Pods After Upgrade
panasonic-qualitycenter-solutionapp-ui-777ff6c44c-72jnm 1/1 Running 0 9s 0.0.54
```

ii. After deployment, validate the deployment status/logs from Jenkins by opening a browser and accessing the URL:
http://<CB_HOSTNAME>:8080/job/panasonic.qualitycenter.solutionapp.ui-pipeline.



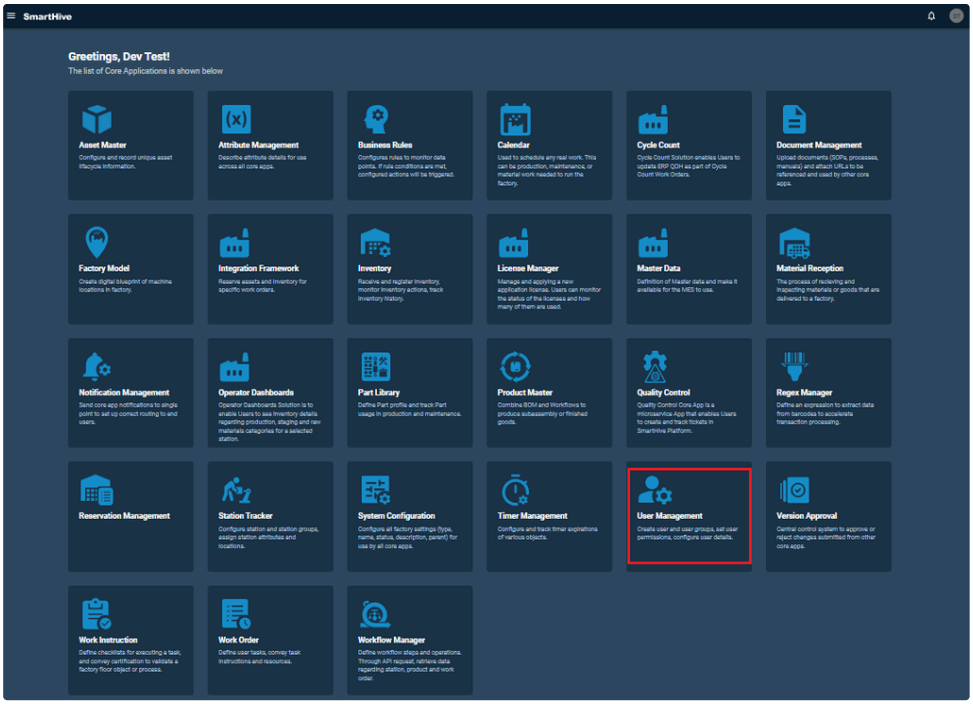
Checking App on the Main/UI

Follow these steps to check app on the main/ui:

1. Open a web browser and enter the URL: https://<K8S_VHOSTNAME>/main/ui.
   The SmartHIVE login page appears.
2. Enter username and password in their respective fields with UI_USER/UI_PWD (from config.ini).

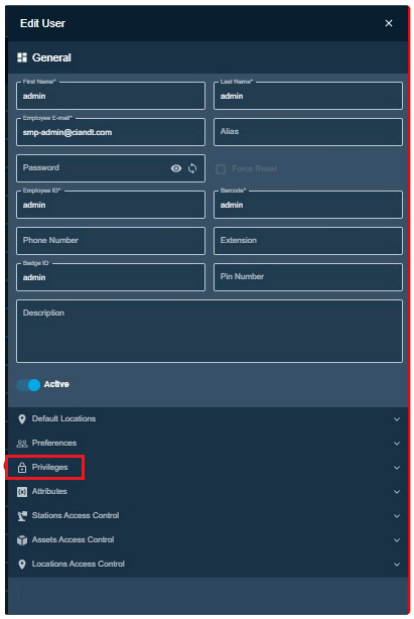
Note: If you have forgotten your password, click the **Forgot Password?** link and follow the instructions to reset your password.

3. Click **LOG IN**.
   The SmartHive Home page appears when you logged in.
4. Give required permissions to the newly added app (not required for upgrades):
   a. Click on the **User Management** application.

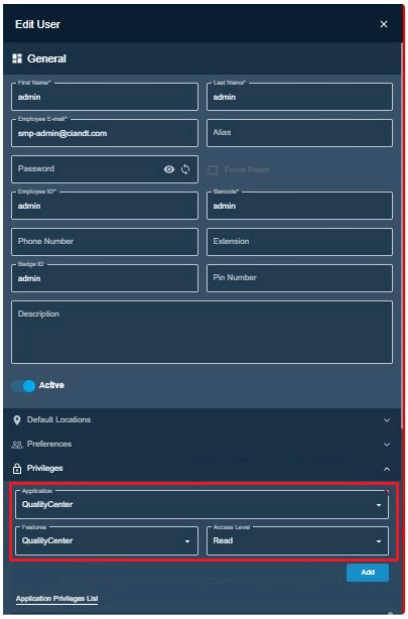


   b. Select a user with admin access.

c. Click on **Privileges**.

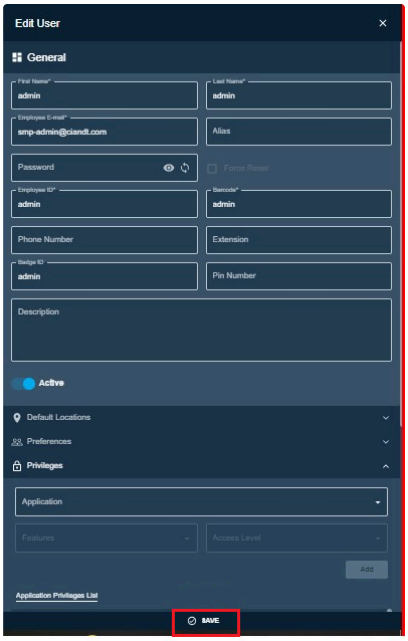


d. Select Application, Features, and Access Level from the drop-down list.

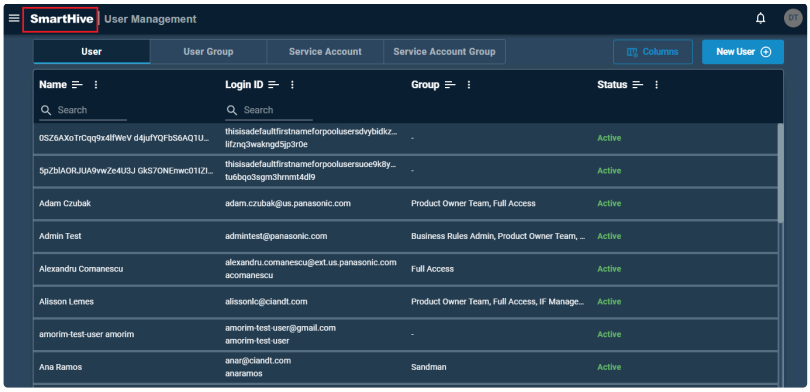


e. Click **Add**.

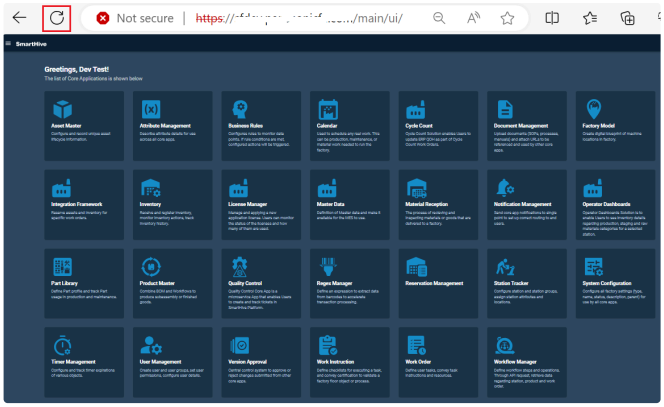
f. Click on **SAVE** icon.



5. Click **SmartHive** to go back to the main/ui page.



6. Click on **Refresh** icon.



The new app will be available on the main/ui page.

Exporting a Snapshot of Settings

Exporting a snapshot of all the settings using Velero when the system works involves creating backups of the Kubernetes cluster resources and persistent volumes. It is crucial for future troubleshooting, replication, and auditing processes. The backup is the full backup of the cluster resources, and the ideal backup frequency is daily.

The Velero backup uploads to the cloud storage account as a file. The next scheduling iteration will handle the backup sync-up if the cloud connectivity has broken.

Note: The Kubernetes cluster components are automatically backed up in the cloud on a daily basis.

Prerequisites

- A working Kubernetes cluster.
- Velero installed and configured on the cluster.
- Sufficient permissions to create backups and access to a storage location (e.g., S3 bucket, Azure Blob Storage, etc.).

Backup Using Velero

Note: Ensure that Velero is configured with the appropriate backup storage location. This typically involves setting up a bucket in S3, Azure Blob Storage, or another supported storage provider.

1. Create a backup of your entire Kubernetes cluster using the following command:

   ```
   velero backup create <backup-name> --include-namespaces '*'
   ```

2. Replace `<backup-name>` with a name for your backup.
   The `--include-namespaces '*'` option ensures that all namespaces are included in the backup.

Note: It is possible to schedule the backup once a day. For example, you can schedule the backup at 02:00 AM local time (time TBD) with the following command:

```
velero schedule create my-scheduled-backup --schedule "0 2 * * *" --default-volumes-to-fs-backup --selector
'openebs.io/component notin (jiva-replica,jiva-controller), backup notin (velero-nobackup)'
```

This backup comprises the whole cluster, except for the resources filtered out in the command. By default, Velero keeps the backup for 30 days. If we want to change the backup TTL, you can do it by adding the --ttl option, (e.g., --ttl 24h0m0s).

3. Check the status of your backup to ensure it completed successfully:

   ```
   velero backup describe <backup-name>
   ```

Restore Using Velero

1. Point `kubectl` to the cluster where it needs to be recovered.
2. Delete the resources related to Documents App in the cluster.

Note: If required, consider deleting other resources (secrets, configmaps, ingress, and virtualservices).

3. Restore the cluster from a backup using the following command:

   ```
   velero restore create --from-backup <backup-name>
   ```

5. Check the status of restore details using the following command:
   `velero restore describe <restore-name>`

> **Note**: If the snapshot process fails or is not feasible, it is essential to document the settings manually or using automated scripts.

Manual Documentation Method

Follow these steps to document the settings manually:

1. Create a comprehensive document using Microsoft Word, Google Docs, or Markdown files to document each setting.

> **Note**: You can include screenshots, command outputs, configuration files, and descriptions.

2. Use a structured format to organize settings by categories.
   Example Structure:
   - **Section 1: Network Settings**
     - IP Addresses
     - Subnet Masks
     - DNS Servers
   - **Section 2: System Settings**
     - Operating System Version
     - Installed Packages
     - System Environment Variables
   - **Section 3: Application Configurations**
     - Application Version
     - Configuration Files
     - Deployment Scripts

Automated Documentation Method

Follow these steps to document the settings automatically:

1. Use `kubectl` commands to export the current state of your Kubernetes resources:

   ```
   kubectl get --ignore-not-found -Aoyaml `kubectl api-resources --verbs=list -oname|paste -sd,` > backup_`date +"%F"`.yml
   ```

> **Note**: To create a backup.yaml with all the resources you must get the list of available resources from `kubectl api-resources` and feed it to `kubectl get` . You can use the following command to export specific resource types, (e.g., deployments): `kubectl get deployments --all-namespaces -o yaml > deployments.yaml`.

2. Create scripts to gather system information and configurations.
   For example, you can use the following script to document network settings:

   ```
   mkdir -p /path/to/documentation
   ifconfig -a > /path/to/documentation/network_interfaces.txt
   netstat -rn > /path/to/documentation/routing_table.txt
   cat /etc/resolv.conf > /path/to/documentation/dns_settings.txt
   ```

3. Store the configuration files in a version control system like Git to keep track of changes over time.

Solution Applications Installation

SmartHive Core App / Support App / Adapter Fresh/Upgrade Deployment Steps

Obtain Deployment Scripts

Access the deployment scripts from Bitbucket, which are crucial for deploying the applications. These scripts contain the necessary instructions and configurations for deployment.

Login to Linux Machine

Log in to the Linux machine that has access to the Customer Control Box. This machine will be used to initiate the deployment process.

Retrieve Code

Retrieving code on the Customer Control Box consists of three files:

- **Apps**: The Apps file is used to list all the applications to be deployed or updated. Each entry should follow the format **<app name>-<version/tag>**. For example, panasonic-qualitycenter-solutionapp-ui-0.0.54. The file serves as a list of applications to be deployed or updated.
- **Config.ini**: Contains customer-specific properties such as CB_HOSTNAME, CB_USERNAME, CB_PASSWORD, K8S_MASTER, K8S_VHOSTNAME, UI_USER, and UI_PWD.
- **Deploy.sh Script**: The Deploy.sh is used to execute the deployment script. The deploy.sh script reads data from apps and config.ini to deploy the apps.

Run Deployment Script

Execute the deployment script using ./deploy.sh and follow the instructions provided. This script automates the deployment process and ensures that all necessary configurations are applied correctly.

Validate Deployment

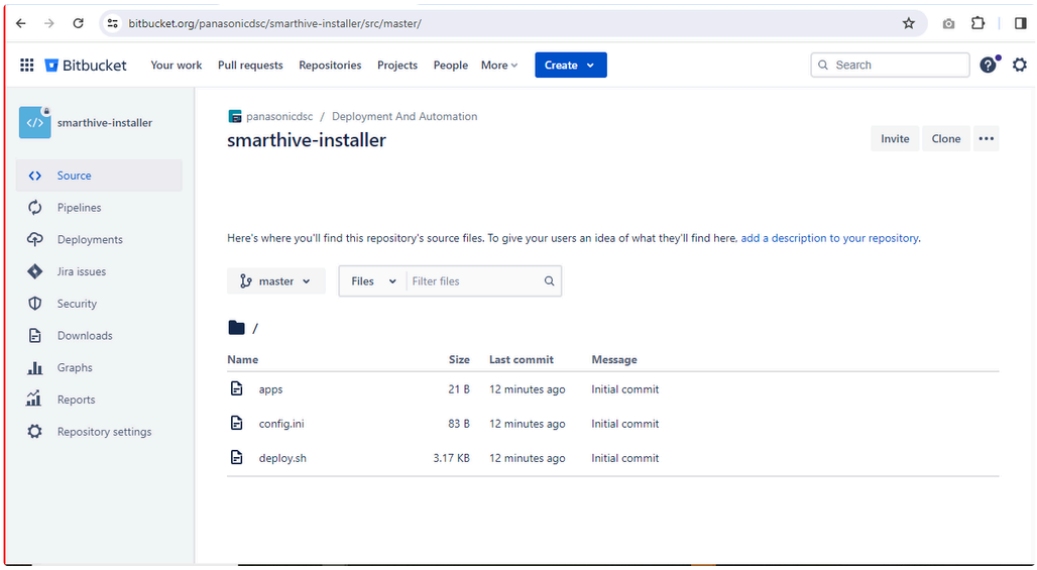
After deployment, validate the deployment status/logs from Jenkins by accessing the specified URL. This step helps ensure that the deployment was successful and allows you to review any logs or errors that may have occurred during the deployment process.

Check Application

Check the deployed application on the main/ui page at the specified URL. Login with the credentials specified in config.ini (UI_USER/UI_PWD).

Deployment of SmartHive Solution Application

1. Obtain the deployment scripts from Bitbucket at the following URL: https://bitbucket.org/panasonicdsc/smarthive-installer/src/master/



2.

a. Login to the Linux machine that has access to the Customer Control Box.

b. Retrieve the code on the Customer Control Box. The code comprises the following three files:

i. **Apps**: Update this file with all the applications to be deployed, following the format **<app name>-<version/tag>**.

For example, panasonic-qualitycenter-solutionapp-ui-0.0.54, where panasonic-qualitycenter-solutionapp-ui is the app name and 0.0.54 is the version/tag.

ii. **Config.ini**: This file contains customer-specific properties, such as:

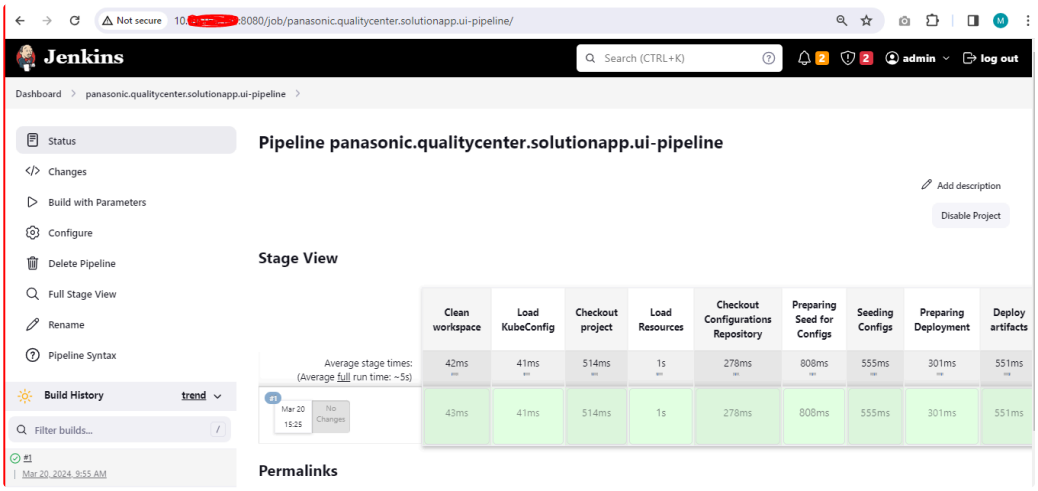
- CB_HOSTNAME
- CB_USERNAME
- CB_PASSWORD
- K8S_MASTER
- K8S_VHOSTNAME
- UI_USER
- UI_PWD

iii. **Deploy.sh:** This script reads data from **apps** and **config.ini** to deploy the apps on the target environment.

1. Run the script using **./deploy.sh** and follow the instructions.

```
cimbuild@infra-ubu-test5:~/app_deployment/customer$ ./deploy.sh
[INFO]: Deploying
   APP: panasonic-qualitycenter-solutionapp-ui
   TAG: 0.0.54
[CHK ]: Press [Y/y] to proceed: y
proceeding ...
[INFO]: Copying panasonic-qualitycenter-solutionapp-ui-0.0.54.tar.gz to <CB_HOSTNAME>
panasonic-qualitycenter-solutionapp-ui-0.0.54.tar.gz 100% 18MB 1.6MB/s 00:11
[INFO]: Extracting Offline Package on <CB_HOSTNAME>
[INFO]: Updating Config.ini
config.ini 100% 188 1.0KB/s 00:00
[INFO]: Pods Before Upgrade
panasonic-qualitycenter-solutionapp-ui-864fd86698-phvzn 1/1 Running 0 48d 0.0.40
[INFO]: Starting Deployment
[INFO]: Login to docker registry...
Login Succeeded
[INFO]: Pushing panasonic-qualitycenter-solutionapp-ui:0.0.54 to docker registry...
[INFO]: Jenkins running at http://<CB_HOSTNAME>:8080/
[INFO]: Triggering Pipeline mwsinv-ingress-single-seed
Build Status: SUCCESS
[INFO]: Triggering Pipeline mwsinv-istio-single-seed
Build Status: SUCCESS
[INFO]: Triggering Pipeline panasonic.qualitycenter.solutionapp.ui-pipeline with Tag
Build Status: SUCCESS
[INFO]: Running App Permissions
[INFO]: Running App Settings
[INFO]: Running App NotificationTrigger
[INFO]: Pods After Upgrade
panasonic-qualitycenter-solutionapp-ui-777ff6c44c-72jnm 1/1 Running 0 9s 0.0.54
```

2. After deployment, validate the deployment status/logs from Jenkins by opening a browser and accessing the URL:
http://<CB_HOSTNAME>:8080/job/panasonic.qualitycenter.solutionapp.ui-pipeline.



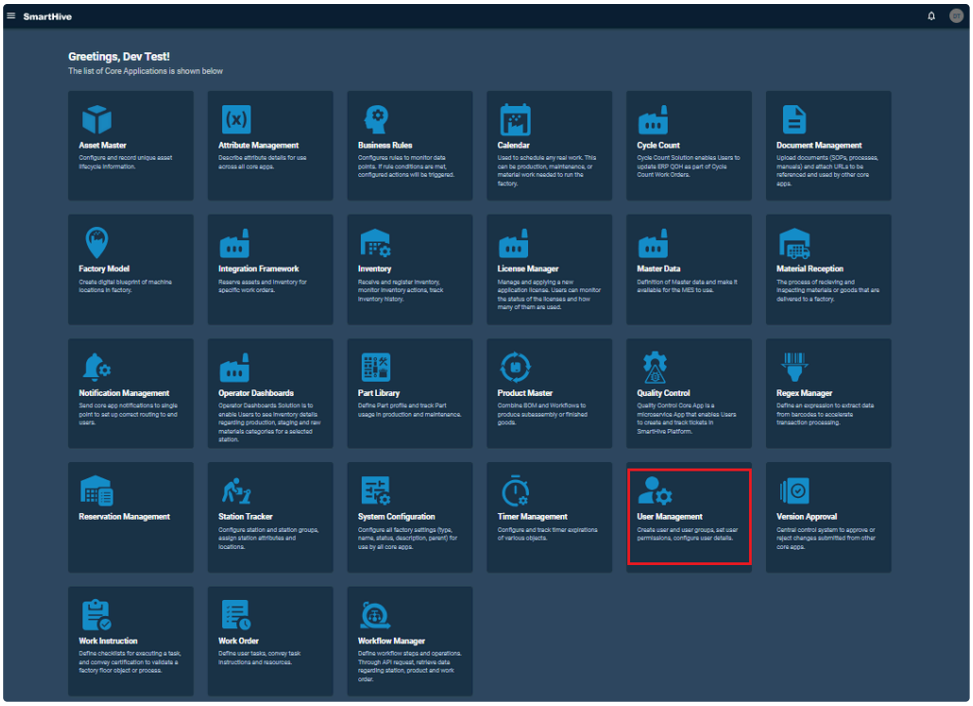
Checking App on the Main/UI

Follow these steps to check app on the main/ui:

1. Open a web browser and enter the URL: https://<K8S_VHOSTNAME>/main/ui.
   The SmartHIVE login page appears.
2. Enter username and password in their respective fields with UI_USER/UI_PWD (from config.ini).

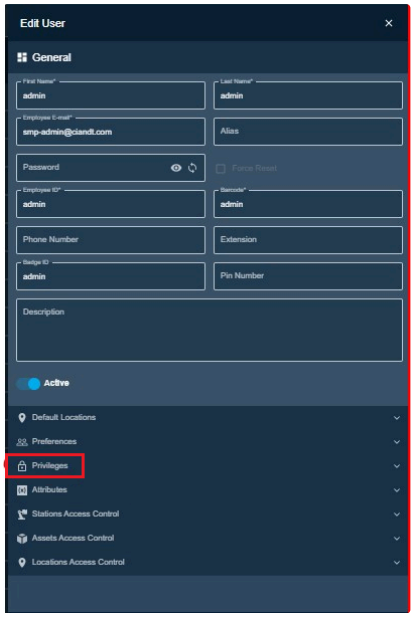
Note: If you have forgotten your password, click the **Forgot Password?** link and follow the instructions to reset your password.

3. Click **LOG IN**.
   The SmartHive Home page appears when you logged in.
4. Give required permissions to the newly added app (not required for upgrades):
   a. Click on the **User Management** application.

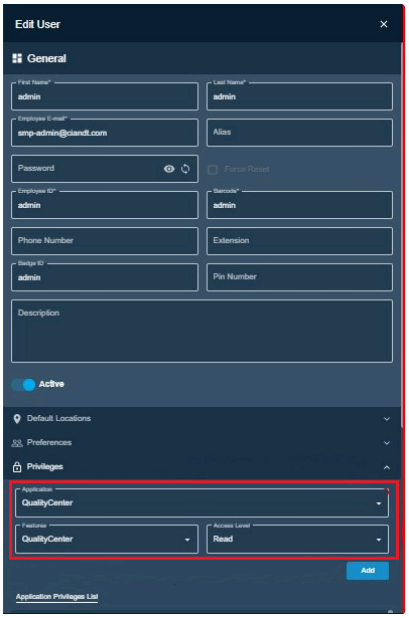


b. Select a user with admin access.

c. Click on **Privileges**.

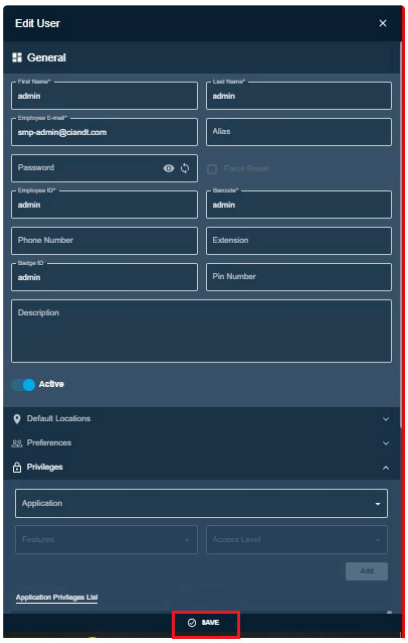


d. Select Application, Features, and Access Level from the drop-down list.

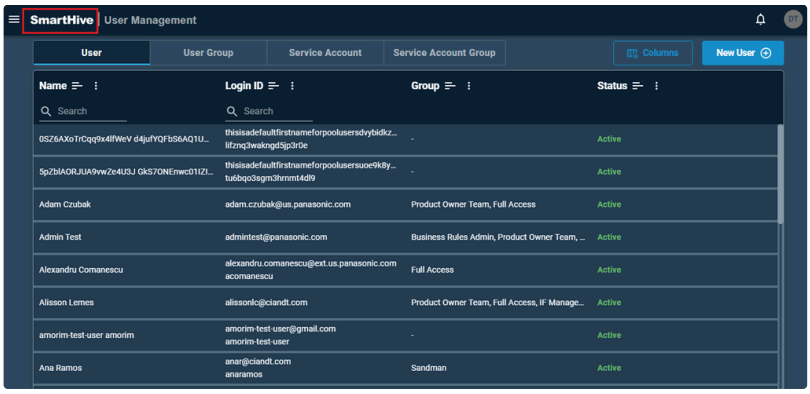


e. Click **Add**.

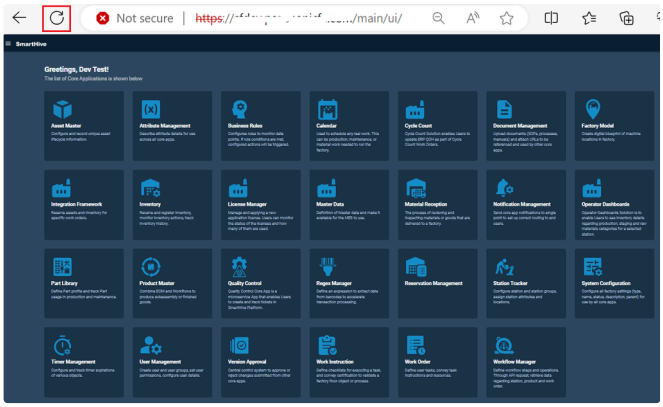
f. Click on **SAVE** icon.



5. Click **SmartHive** to go back to the main/ui page.



6. Click on **Refresh** icon.



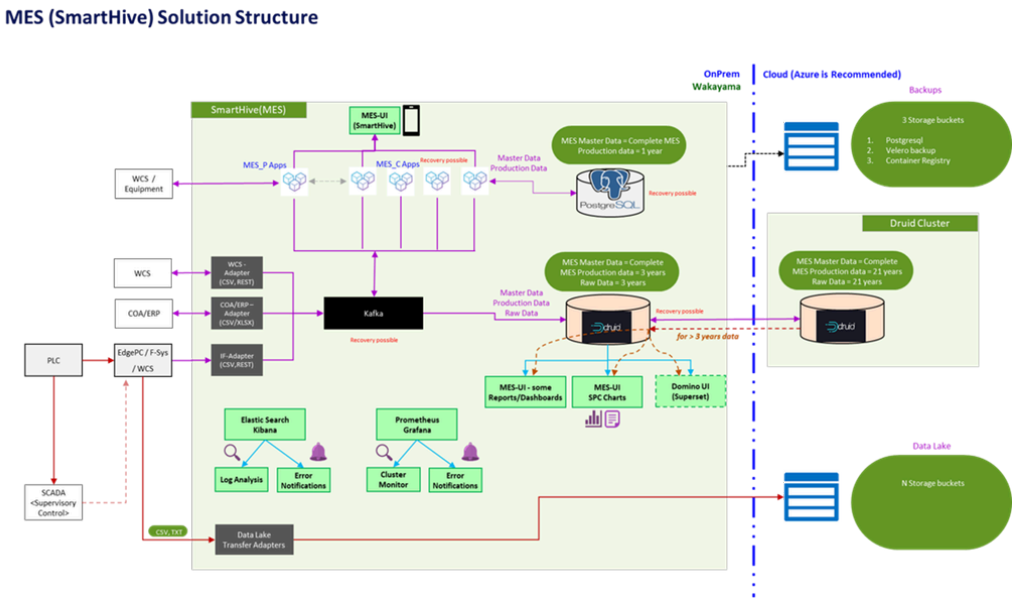
The new app will be available on the main/ui page.

Solution Map

The solution map provides a comprehensive overview of the MES (SmartHive) solution architecture, highlighting the flow of data from various industrial sources through processing and storage systems, to user interfaces and monitoring tools. The architecture ensures data integrity, accessibility, and resilience through integrated messaging, storage, and backup solutions.

> **Note**: It is recommended to use Microsoft Azure as the Cloud Vendor. Because, all backup operations, scripts, and Velero tools are verified with Azure cloud only.

The following is the MES (SmartHive) Solution Structure with connected devices:



Key Components and Data Flow

- **SmartHive (MES)**:
  - Core of the system, interacting with various components and providing the main user interface (MES-UI).
- **Data Sources**:
  - **WCS/Equipment**: Interfaced with MES_P Apps.
  - **WCS, COA/ERP**: Connected via adapters (CSV, REST, XLSX).
  - **PLC, EdgePC/F.Sys, WCS**: Collecting data through IF-Adapters (CSV, REST) and sending to Kafka.
  - **SCADA (Supervisory Control)**: Sends data (CSV, TXT) to EdgePC/F.Sys and WCS.
- **Messaging and Data Processing**:
  - **Kafka**: Central messaging system ensuring data is distributed to the appropriate services, with recovery mechanisms in place.
- **Data Storage**:
  - **PostgreSQL**: Stores MES Master Data and Complete Production Data for 1 year.
  - **Druid**: Stores MES Master Data and Complete Production Data for longer terms (3 years for MES Master Data, up to 21 years for production data).
- **Analytics and Monitoring**:
  - **Elastic Search/Kibana**: For log analysis and error notifications.
  - **Prometheus/Grafana**: For cluster monitoring and error notifications.

- **User Interfaces and Reports**:
  - **MES-UI**: Provides some reports and dashboards.
  - **MES-UI SPC Charts**: Specific for Statistical Process Control charts.
  - **Domino UI (Superset)**: Advanced reporting interface for extended data analysis.
- **Backup and Data Lake**:
  - **Cloud Backup**: Utilizes three storage buckets for PostgreSQL, Velero backup, and Container Registry.
  - **Data Lake**: Long-term storage for extensive datasets, facilitated by data lake transfer adapters.

Integration of MES (SmartHive) Solution Structure

- Data from various sources (WCS, PLC, COA/ERP, SCADA) is aggregated and processed by SmartHive (MES) through adapters and Kafka.
- Kafka distributes the data to storage solutions (PostgreSQL and Druid) and monitoring tools (Elastic Search/Kibana and Prometheus/Grafana).
- User interfaces (MES-UI, MES-UI SPC Charts, Domino UI) provide access to processed data for reporting and analysis.
- Backup mechanisms ensure data resilience and long-term storage is managed by Druid clusters and the data lake.

The Workflow Manager

Workflow

A workflow is a sequence of tasks that represents a process or activity on the factory floor which can help you standardize and streamline task execution. In SmartHive, a workflow enables you to visually model business processes. You can then streamline or automate these processes by setting up triggers across applications to which you have access.

Workflow functions as a generic execution engine, capable of interacting with any element in the MES platform. This means that workflows can be tailored for various entities, including Material, Product, Station, and more. Moreover, they have the ability to execute custom logic as required. Users have the capacity to monitor and guide any object (such as Material, Product, Carrier, etc.) through a workflow, allowing for comprehensive control and execution.

A workflow in SmartHive typically involves the following elements:

1. **Tasks:** Individual steps or actions that need to be performed during the process.
2. **Triggers:** Events or conditions that initiate the workflow. These could be actions performed by users, a time-based trigger, or the occurrence of certain data-related events.
3. **Automation:** Predefined rules or logic that govern how the tasks are executed based on triggers and other relevant factors.
4. **Notifications:** Alerts or messages that can be sent to relevant parties to keep them informed about the progress or completion of workflows.
5. **Approvals:** Steps associated with tasks that require authorization from other users.

Workflow Objects

The Workflow Manager represents each aspect of a manufacturing operation as an object. Each of these objects has its own data model, and one or more state models. Each of these models can be extended using custom data fields. Additionally, objects can interact with each other and carry a history of these interactions throughout the lifecycle of the process.

Here are some examples of frequently used workflow objects:

- **WIP Material:** A WIP (work in progress) material is an object that represents a product that’s in the process of being manufactured. For example, consider a circuit board moving down an assembly line, eventually becoming a laptop or a phone. General use cases involve tracking the WIP through each step, controlling routing, and making process decisions to control the manufacturing operation. These decisions are made in real time and involve everything from preparation, processing, and quality control.
- **Material (MAT):** The term Material refers to raw material/goods that are purchased and used to assemble--or add to--another WIP Material during manufacturing. General use cases involve tracking material movements, who moved them, how much quantity is available, where it is currently located, receiving/incoming quality decisions, etc.
- **Defect (DEF):** A defect or problem with any object. Defects can be anything from a problem with a station, a product, a material, or a user. It applies to any other object to indicate that:
  a. There is a problem.
  b. This is the problem.
  c. This is what was done to fix it.
  d. It is no longer a problem.
- **Station (STA):** Represents physical workplaces on the factory floor, and serves as a single point of representation and control of a work area.

Workflow Manager

Workflow Manager enables you to manage workflows and orchestrate the flow of data and tasks across applications in SmartHive.

The primary objective of the Workflow Manager is twofold:

1. **Automate Business Logic and Processes**: The primary objective of Workflow Manager is to automate business logic and processes within the SmartHive platform. It involves mapping out the sequence of steps, decision points, inputs, outputs, and interactions between different elements of the manufacturing process.
2. **Support for External Parties**: Workflow functionality is designed not only for internal use but also to serve as an engine that external parties can utilize to automate their own business processes.

To achieve these objectives, the Workflow Manager application enables users to model complex business logic by utilizing basic building blocks, each equipped with functions like API calls, message publication, sequencing, and routing. By utilizing basic building blocks and functions, Workflow Manager enables users to create, customize, and automate their business processes efficiently.

Adapters Installation

SmartHive Core App / Support App / Adapter Fresh/ Upgrade Deployment Steps

Obtain Deployment Scripts

Access the deployment scripts from Bitbucket at the following URL: Bitbucket Repository URL. These scripts are essential for deploying the applications on the Customer Control Box.

Login to Linux Machine

Log in to the Linux machine that has access to the Customer Control Box. Ensure you have the necessary permissions and credentials.

Retrieve Code

Retrieve the code on the Customer Control Box, which consists of three files:

- **Apps:** Update this file with all the applications to be deployed, following the format <app name>-<version/tag>.
- **config.ini:** Contains customer-specific properties such as CB_HOSTNAME, CB_USERNAME, CB_PASSWORD, K8S_MASTER, K8S_VHOSTNAME, UI_USER, and UI_PWD.
- **deploy.sh:** This script reads data from apps and config.ini to deploy the apps on the target environment.

Run Deployment Script

Execute the deployment script using ./deploy.sh and follow the instructions provided. The script will start the deployment process, including pulling the latest code from the repository, building the applications, and deploying them to the target environment. Ensure the script completes successfully without errors.

Validate Deployment

After deployment, validate the deployment status/logs from Jenkins by accessing the URL: Jenkins URL. This step ensures the deployment was successful and the applications are running correctly in the target environment.

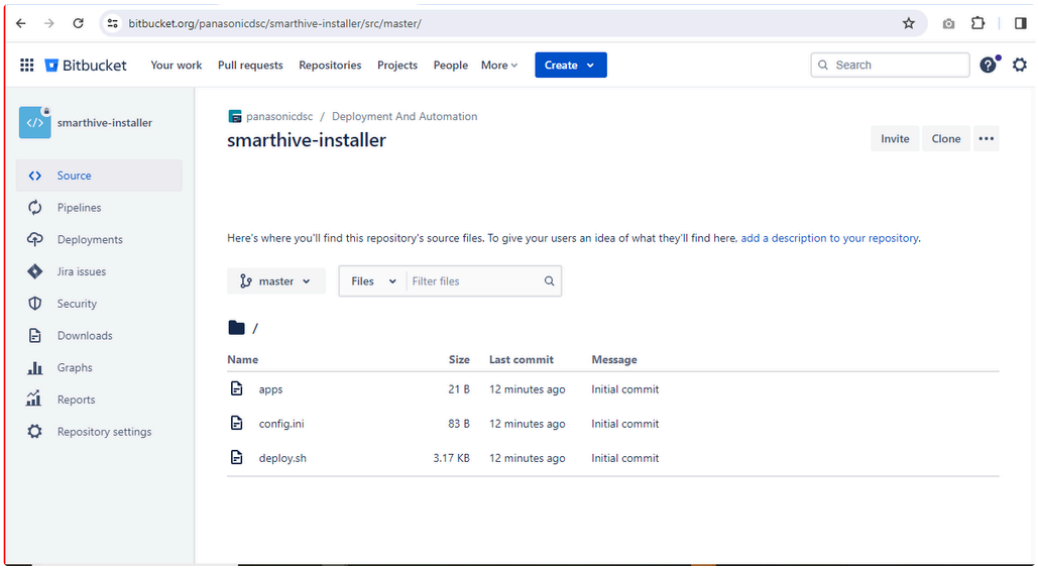
Check Application

Check the application on the main/ui page at Main/UI Page URL. Login with the credentials specified in config.ini (UI_USER/UI_PWD). Give required permissions to the newly added app (not required for upgrades) by navigating to user management, selecting a user with admin access, clicking on 'Privileges', selecting Application, Features, and Access level, and then clicking on SAVE button. Finally, refresh the main/ui page to see the new app available on the main/ui page.

Step by step instructions for SmartHive Adapters Application

1. Obtain the deployment scripts from Bitbucket at the following URL:

https://bitbucket.org/panasonicdsc/smarthive-installer/src/master/



2. Login to the Linux machine with access to the Customer Control Box.
3. Retrieve the code on the Customer Control Box. The code comprises three files:

- **Apps**: Update this file with all the applications to be deployed, following the format **<app name>-<version/tag>**. For example:

    i. panasonic-qualitycenter-solutionapp-ui-0.0.54

    ii. panasonic-qualitycenter-solutionapp-ui is app name

    iii. 0.0.54 is version/tag

- **config.ini:** Contains customer-specific properties such as:

The file contains the customer specific properties

i. CB_HOSTNAME

ii. CB_USERNAME

iii. CB_PASSWORD

iv. K8S_MASTER

v. K8S_VHOSTNAME

vi. UI_USER

vii. UI_PWD

- **Deploy.sh:** This script reads data from apps and **config.ini** to deploy the apps on the target environment.

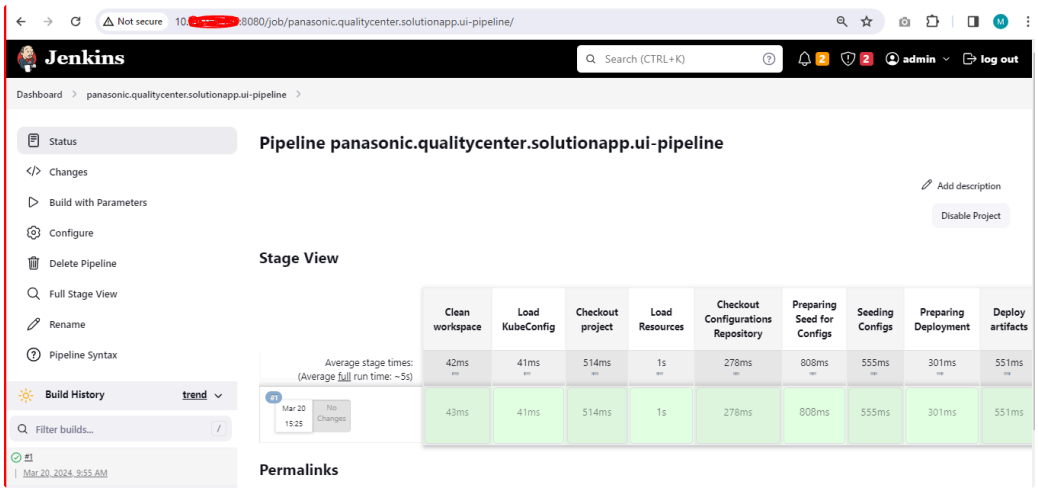
4. Run the script using

./deploy.sh and follow the instructions.

```
cimbuild@infra-ubu-test5:~/app_deployment/customer$ ./deploy.sh
[INFO]: Deploying
   APP: panasonic-qualitycenter-solutionapp-ui
   TAG: 0.0.54
[CHK ]: Press [Y/y] to proceed: y
proceeding ...
[INFO]: Copying panasonic-qualitycenter-solutionapp-ui-0.0.54.tar.gz to <CB_HOSTNAME>
panasonic-qualitycenter-solutionapp-ui-0.0.54.tar.gz 100% 18MB 1.6MB/s 00:11
[INFO]: Extracting Offline Package on <CB_HOSTNAME>
[INFO]: Updating Config.ini
config.ini 100% 188 1.0KB/s 00:00
[INFO]: Pods Before Upgrade
panasonic-qualitycenter-solutionapp-ui-864fd86698-phvzn 1/1 Running 0 48d 0.0.40
[INFO]: Starting Deployment
[INFO]: Login to docker registry...
Login Succeeded
[INFO]: Pushing panasonic-qualitycenter-solutionapp-ui:0.0.54 to docker registry...
[INFO]: Jenkins running at http://<CB_HOSTNAME>:8080/
[INFO]: Triggering Pipeline mwsinv-ingress-single-seed
Build Status: SUCCESS
[INFO]: Triggering Pipeline mwsinv-istio-single-seed
Build Status: SUCCESS
[INFO]: Triggering Pipeline panasonic.qualitycenter.solutionapp.ui-pipeline with Tag
Build Status: SUCCESS
[INFO]: Running App Permissions
[INFO]: Running App Settings
[INFO]: Running App NotificationTrigger
[INFO]: Pods After Upgrade
panasonic-qualitycenter-solutionapp-ui-777ff6c44c-72jnm 1/1 Running 0 9s 0.0.54
```

5. After deployment, validate the deployment status/logs from Jenkins by opening a browser and accessing the URL:

http://<CB_HOSTNAME>:8080/job/panasonic.qualitycenter.solutionapp.ui-pipeline.



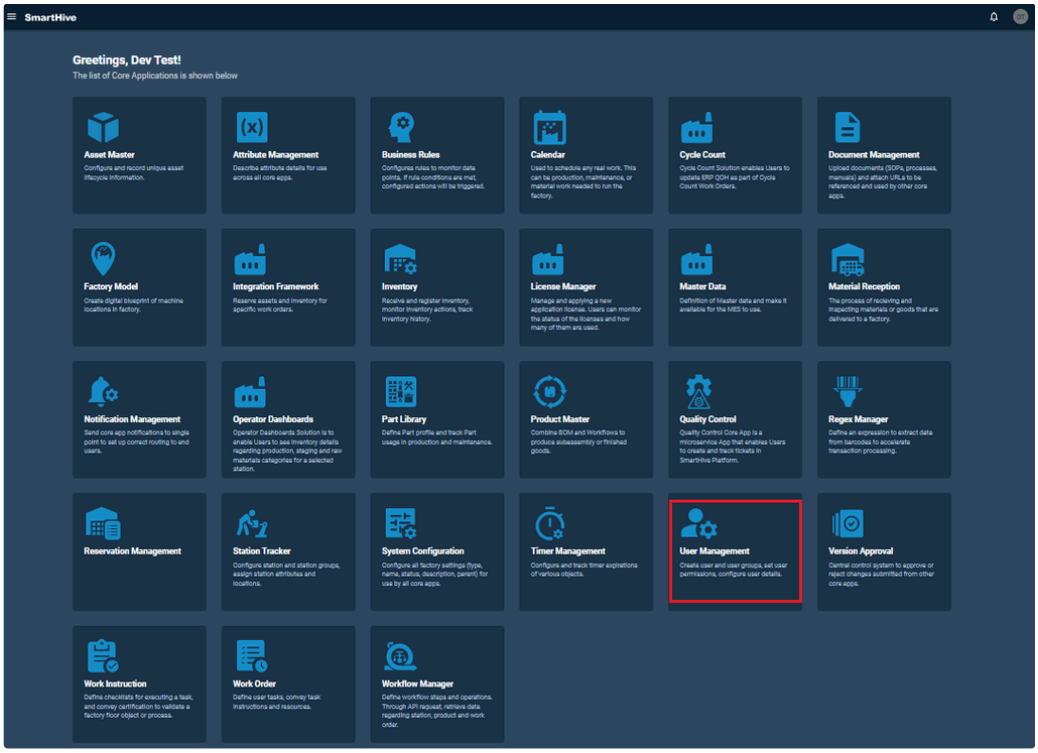
6. Check the app on the main/ui page at

https://<K8S_VHOSTNAME>/main/ui.

Login with **UI_USER/UI_PWD (from config.ini)**

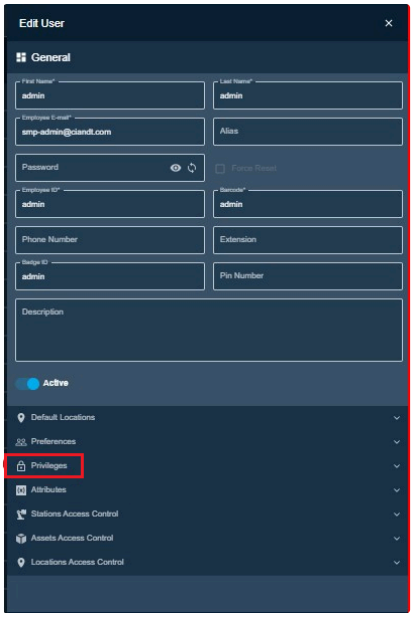
Give required permissions to the newly added app (not required for upgrades):

1. Click on the **User Management** application.

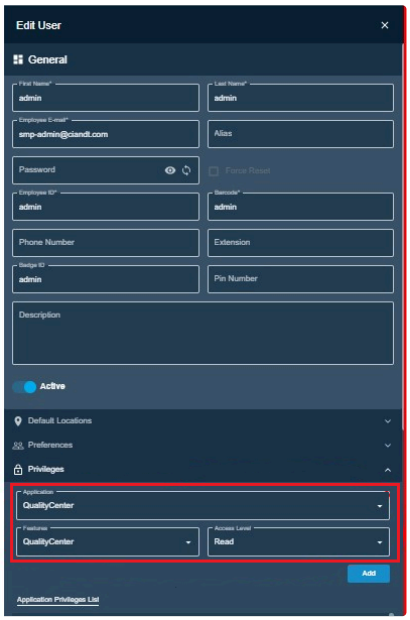


2. Select a user with admin access.

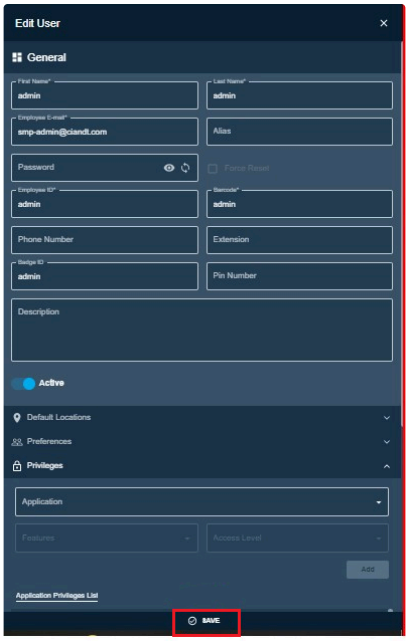
3. Click on **Privileges**.



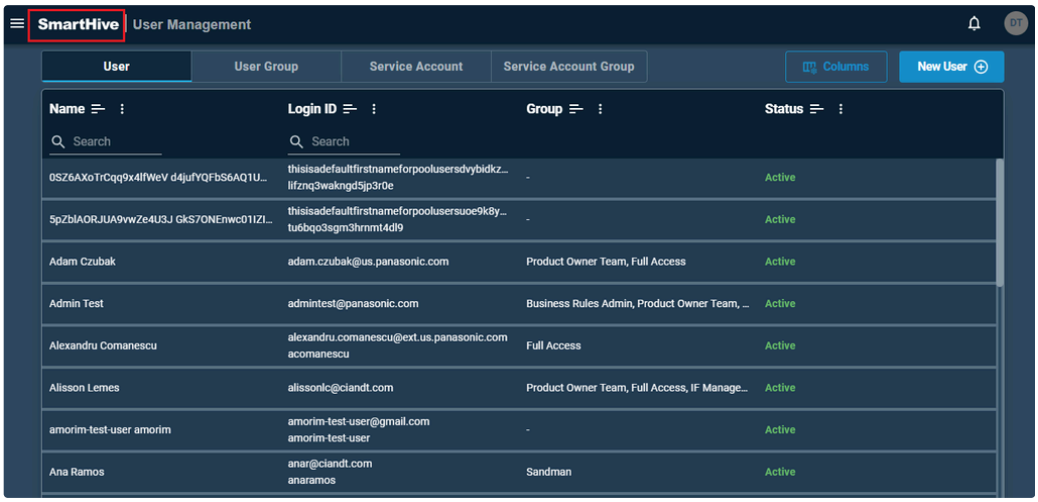
4. Select Application, Features, and Access Level from the drop-down list, and click **ADD**.



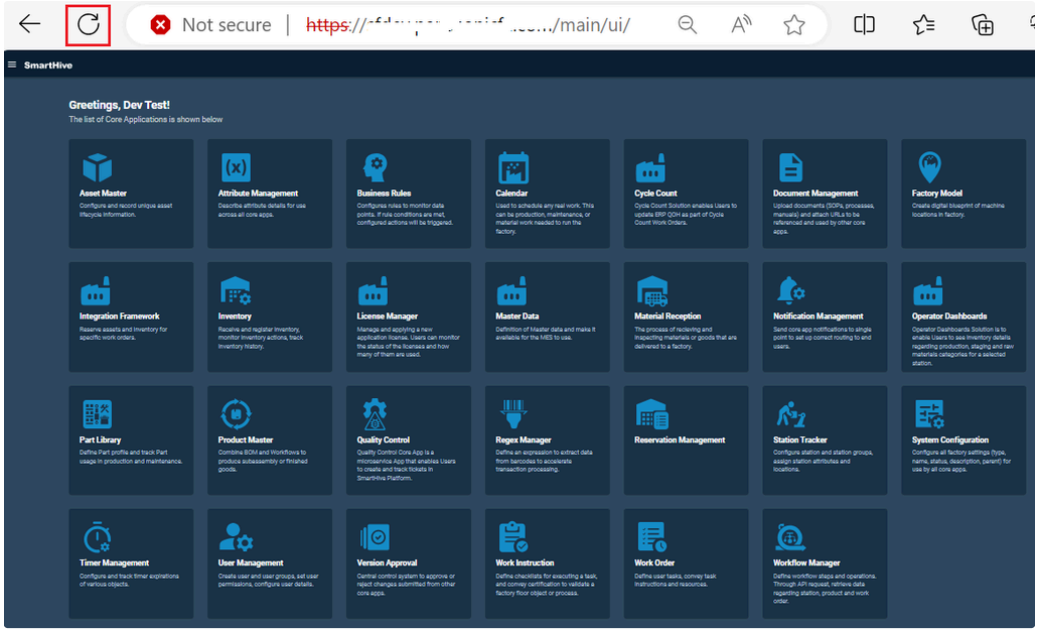
5. Click on **SAVE** icon.



6. Click **SmartHive** to go back to the main/ui page.



7 Click on **Refresh** icon.



The new app will be available on the main/ui page.

Upgrades

Upgrading Applications

Upgrading applications in a Kubernetes environment follows a similar process to the initial installation of Infrastructure Installation, Core / Platform Application, and Solution Application. When a new version of an application becomes available, you can upgrade your existing deployment by following these steps:

Check Compatibility

Before upgrading, ensure that the new version of the application is compatible with your current Kubernetes cluster and any dependencies.

Backup Data

It's essential to back up any critical data associated with the application before upgrading to avoid data loss in case of any issues during the upgrade process.

Update Configuration Files

Modify the configuration files of your Kubernetes deployment to specify the new version of the application you want to deploy.

Deploy Upgrade

Use the Kubernetes CLI or dashboard to deploy the updated configuration files, triggering the upgrade process. Kubernetes will manage the rollout of the new version, ensuring minimal downtime and a smooth transition.

Monitor Upgrade

Monitor the upgrade process using Kubernetes monitoring tools to ensure that the new version is deployed correctly and is functioning as expected.

Rollback (if necessary)

If any issues arise during the upgrade process, Kubernetes allows you to roll back to the previous version of the application quickly. This rollback feature helps mitigate risks and ensures that your application remains operational.

System Configuration / Onboarding

System configuration and onboarding are vital for ensuring the reliability and effectiveness of a new system. By following this, organizations can achieve a seamless transition, minimize downtime, and ensure that users are well-supported throughout the process.

Data Backup and Restore

Data backup and restore are critical processes to ensure data integrity, availability, and recovery in case of data loss, corruption, or disasters. The purpose is to recover the Kubernetes cluster components.

> **Note**: The Kubernetes cluster components are automatically backed up in the cloud on a daily basis.

Pre-requisites

- Access to Kubernetes cluster.
- Access to Azure storage account.
- Access to the control box where terminal commands are executed.

Dependencies

- Kubernetes admin access and experience with managing SmartHive clusters.
- Fully air-gapped environments (zero cloud access) are not supported for recovery.

Restoring Kubernetes Cluster Components

This procedure describes the process of restoring Kubernetes cluster components.

The restoration process comprises three elements, and it should proceed in the following order:

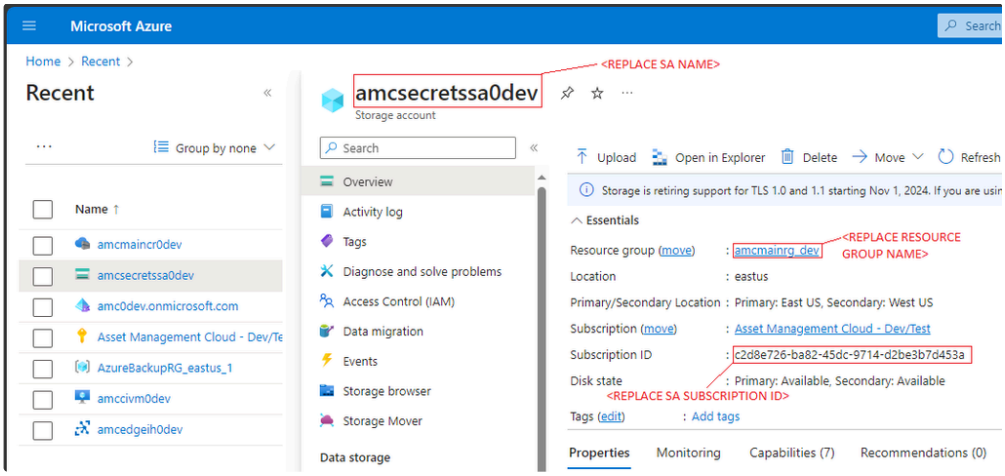
- Local Container Registry (LCR)
- Database
- Kubernetes

Locating Storage Account Metadata

> **Note**: The service engineers authorized for Azure only need to retrieve the information.

Follow these steps to locate the storage account metadata in Microsoft Azure:

1. Open your web browser and go to the Microsoft Azure portal.
2. Sign in with your email and password or select an account to continue to Microsoft Azure.
3. In the Azure portal, search for "Storage accounts" in the search bar at the top.



4. Locate and click on the storage account. In the example, the storage account is named `amcsecretssa0dev`. Once the storage account is selected, you will be directed to its "Overview" page.

5. Check the following fields are updated along with their information in the "Essentials" section:
   - Resource group
   - Subscription
   - Subscription ID
6. Check the "Disk state" field to ensure it shows "Primary: Available, Secondary: Available".
7. Check that the "Tags" field is updated to ensure proper categorization and management of your storage account.

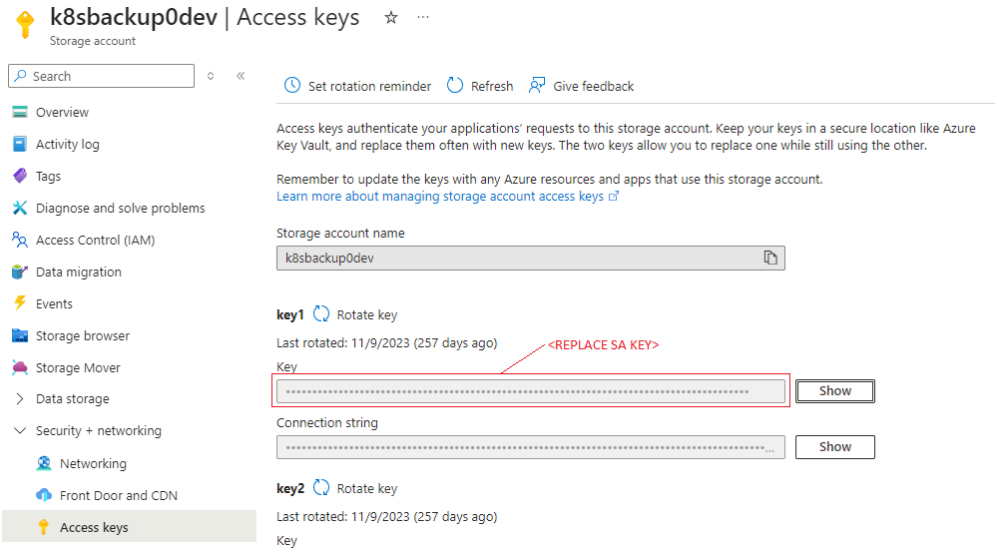
> The tags can be updated in the "Tags" section located in the left menu. Tags must be added or modified to improve categorization and manage the storage account.

8. Review the metadata to confirm all information is correct.

Locating Storage Account Access Key Information

Follow these steps to locate the storage account access key information in Microsoft Azure:

1. Open your web browser and go to the Microsoft Azure portal.
2. Sign in with your email and password or select an account to continue to Microsoft Azure.
3. In the Azure portal, search for "Storage accounts" in the search bar at the top.



4. Locate and click on the storage account. In the example, the storage account is named `k8sbackup0dev`.
5. In the left-hand menu, go to **Security + networking > Access keys**.
   On the "Access keys" page, the current access keys for the storage account are displayed. The page displays two keys (`key1` and `key2`) along with their connection strings.

> **Note**: The connection string will update automatically when the key is added. If necessary, click the "Show" button to verify the key or connection string.

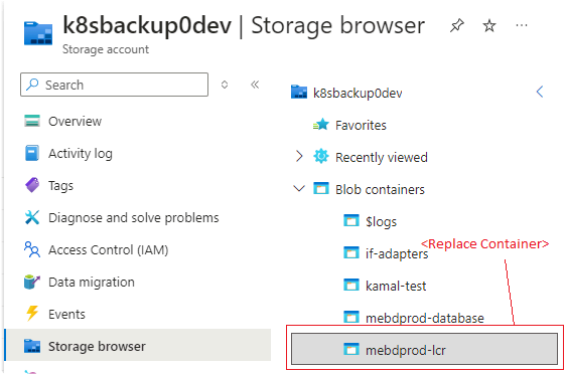
6. Review the key information to confirm all details are correct.

Local Container Registry

The LCR is the Docker registry that contains all the images used by the apps deployed through Kubernetes. It is essential for managing and storing Docker images within a private infrastructure, providing a higher level of control and security over your container images.

Follow these steps to restore the LCR:

1. In the left-hand menu, click Storage Browser.



2. Locate the name of the container. In the example, the container is named `mebdprod-lcr`.
   The backup being restored is one day old.
3. To restore the LCR, run the following command:
   `sudo az storage blob download-batch --overwrite -d '/mnt/registry/' --pattern "*" -s <Replace Container> --account-name <REPLACE SA NAME> --account-key '<REPLACE SA SUBSCRIPTION ID>'`

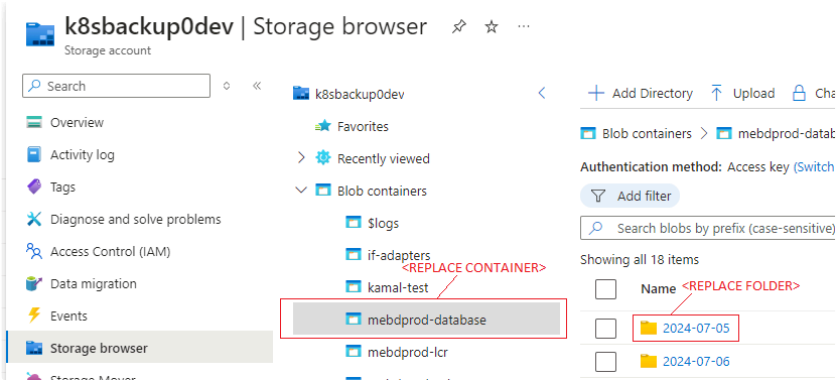
> This command will overwrite the LCR with the cloud version, restoring any missing or corrupt files.

Database

The database server is a PostgreSQL relational database. Files are stored in the cloud and compressed using the xz format. Daily backups of the entire server are stored in separate folders in the YYYY-MM-DD format.

Follow these steps to restore the database:

1. In the left-hand menu, click Storage Browser.



2. In the Storage browser, under the Blob containers section, locate and select the container. In the example, the container is named `mebdprod-database`.
3. Inside the `mebdprod-database` container, locate the folder with the backup date you need to restore. For example, `2024-07-05` or `2024-07-06`.
4. To restore the database, run the following command:

```
# Set up DB connection.
nohup kubectl port-forward service/sm-main-kubegres -nsm-kubegres 5432:5432 --address 0.0.0.0 &>/dev/null & disown
export PGUSER=postgres PGPASSWORD=Panasonic@123 PGDATABASE=postgres PGHOST=localhost PGPORT=5432
# Download database.
az storage blob download-batch --overwrite -d '.' --pattern "<REPLACE SA FOLDER>/*" -s <REPLACE CONTAINER> --account-name <REPLACE SA NAME> --account-key '<REPLACE SA SUBSCRIPTION ID>'
# For each database: [1] decompress file [2] close connections [3] restore db [4] delete file.
```

```
for db in *.xz; do xz -d $db && psql -c 'SELECT pg_terminate_backend(pg_stat_activity.pid) FROM pg_stat_activity WHERE pid <> pg_backend_pid();' && pg_restore --jobs=8 --clean --if-exists --create --dbname=postgres ${db/.xz/} && rm ${db/.xz/}; done
```

This command will overwrite the existing database with the cloud version.

Kubernetes

Kubernetes manifests and resources are backed up using a third-party tool called Velero. The backups are stored as binary data.

To restore the Kubernetes, run the following command:

```
velero restore create --from-backup `velero get backups|awk 'FNR==2{print $1}'` --allow-partially-failed
```

This command will only replace missing resources with the cloud version and will use the latest backup, which is one day old.

Appendix

S.No.	Words	Definitions
1	Docker	Docker is a set of platform-as-a-service products that utilize OS-level virtualization to deliver software in containers.
2	Docker Container	Containers are independent from one another, containing their software, libraries, and configuration files; they can communicate with each other through defined channels.
3	ESXi	VMware ESXi is an enterprise-class, type-1 hypervisor developed by VMware for deploying and serving virtual computers. VMware ESXi is an enterprise-class, type-1 hypervisor developed by VMware for deploying and serving virtual computers.
4	ESXi Virtual Machine	ESXi Virtual Machine emulates a computer system. Virtual machines are based on computer architectures and provide the functionality of a physical computer.

IF Adapters

An Adapter, similar to a "Class" in object-oriented programming, functions as a solution for connecting external systems to SmartHive (SH). It acts as an intermediary, enabling point-to-point connections. Adapters have predefined fields and configurations, serving as templates. However, they do not operate directly; but require a Connection to function.

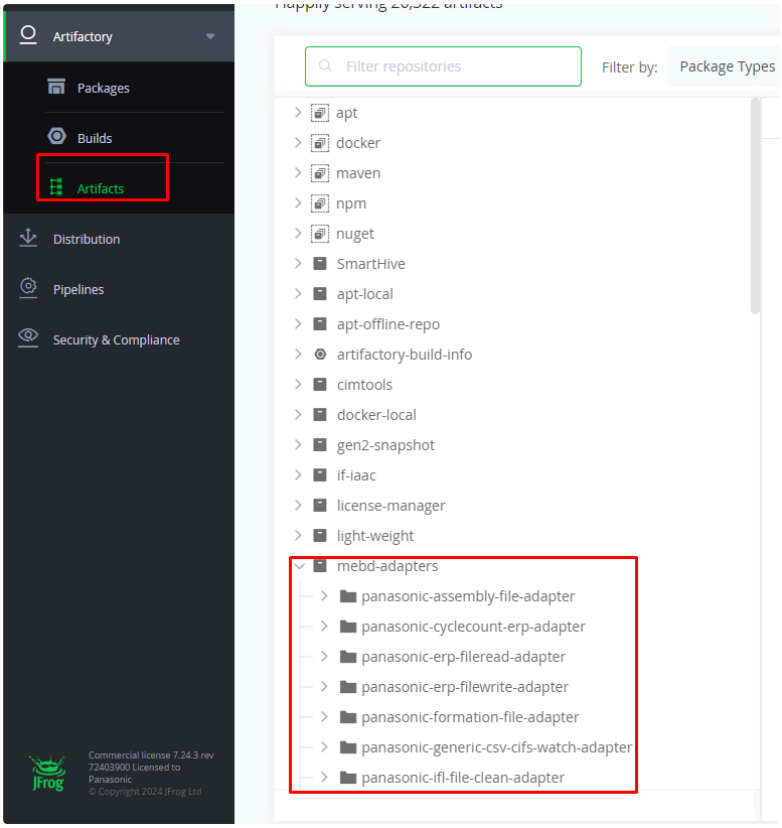
It converts data formats from one system to another, ensuring compatibility and smooth data flow. Also, it supports various communication protocols (e.g., HTTP, FTP, MQTT) to connect different systems. It helps to automate data transfer processes, reducing manual intervention and enhancing efficiency.

The IF adapter is a docker image designed to run within the Linux operating system, as seen from an Integration Framework perspective. This approach offers increased flexibility for adapter development teams, allowing them to utilize any programming language that aligns with their skills. If a component can be dockerized and executed in a Linux environment, it becomes manageable using the Integration Framework. For the list of IF adapters, refer IF Adapter | List of IF Adapters.

Note: Data contracts are crucial in ensuring that data exchanged between systems via IF adapters is consistent, predictable, and correctly processed. By clearly defining the structure, types, constraints, and validation rules in a data contract, you can reduce integration errors and improve the reliability of your workflows. For more details about the Data Contract Format, refer to IF Adapter | Data Contract Format IF Adapters .

Pre-requisites for Setting up the Adapter

- For Adapters:
  - Link to download Adapter: JFrog - http://10.140.64.18:8082/ui.
  - JFrog can only be accessed through a VPN. For account access, please contact the DevOps Team.
  - After logging in to JFrog, you can find the Adapter in Artifacts > mebd-adapters as shown in the image below:



Note: The adapter can only be downloaded from JFrog or imported from a local PC.

- For Connections:
  - To download the connection configuration file, refer to https://panasonic-dsc.atlassian.net/wiki/spaces/SMPUM/pages/442662938/IF+Adapter#Connection-Configuration-Files.

Login to SmartHive

Follow these steps to login to the SmartHive dashboard:

1. Open a web browser and enter the URL: https://<customer domain name>/main/ui to access the SmartHive (e.g. https://sfdev.panasonicfa.com/main/ui).
   The SmartHive login page appears.
2. Enter your username or email ID and password in the respective fields.


3. Click **LOG IN**.
   The SmartHive dashboard opens.

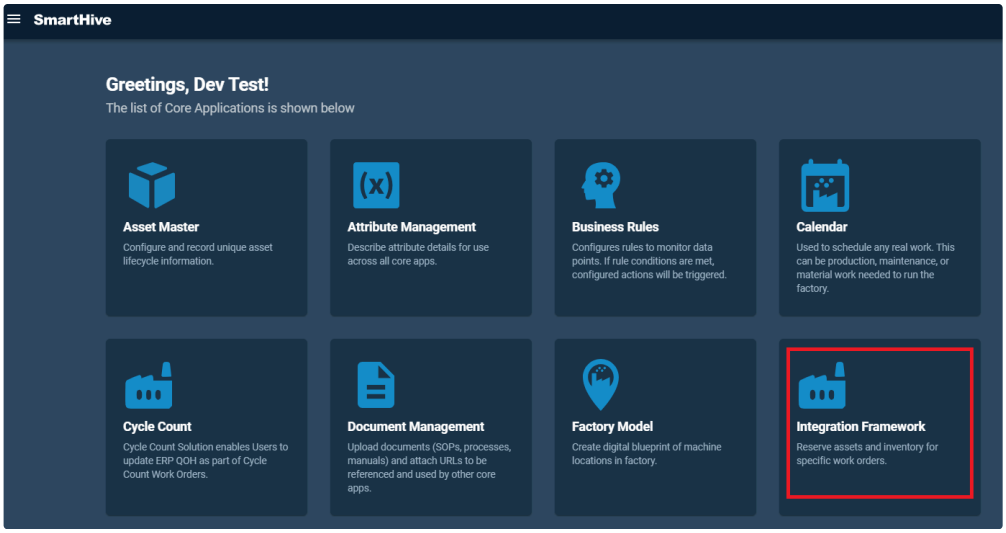
Importing IF Adapters

Importing IF adapters allows you to restore configurations from a backup or migrate them to a new environment.

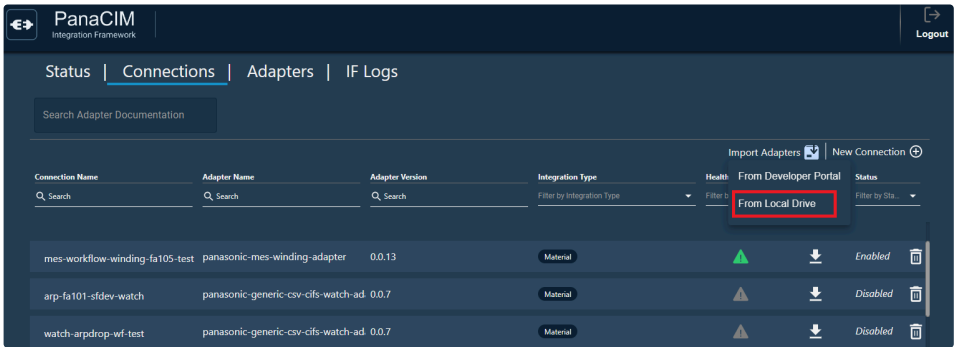
Note: Only adapters with a higher version than the maximum version of the current adapter in the system can be imported.

Follow these steps to import IF adapters:

1. Login to the SmartHive dashboard, see Log In.
2. Click on the **Integration Framework** tile.
   The IF Manager interface opens.

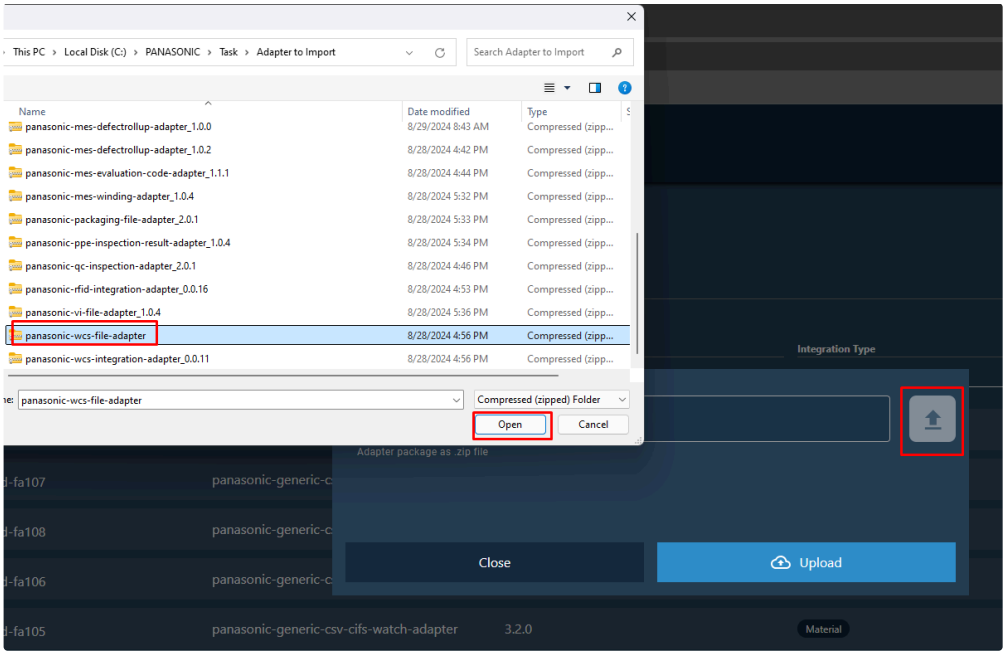


3. Click on the **Connections** tab where you can manage adapters.
4. Click **Import Adapters** and select **From Local Drive** from the list.

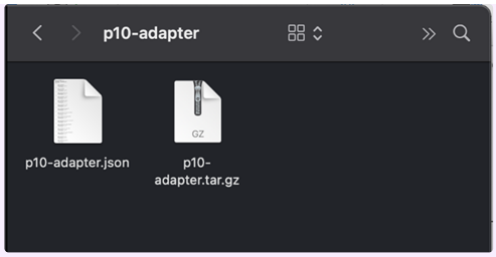


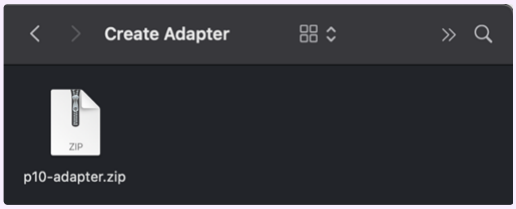
Note: In a cloud environment, there is no option to import from a local drive.

5. Click upload icon to browse and select the zip file from your local machine or backup storage.

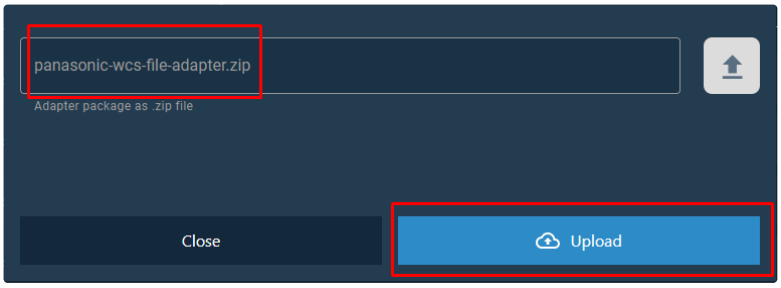


Note: Ensure that both *.tar.gz* and *JSON* files are compressed into a .zip file with the same name and are kept at the root level of the zip file rather than in a subfolder. The name of the zip file must be identical to the names of the *.tar.gz* and *JSON* files. Uploading will result in an error if the names are different.



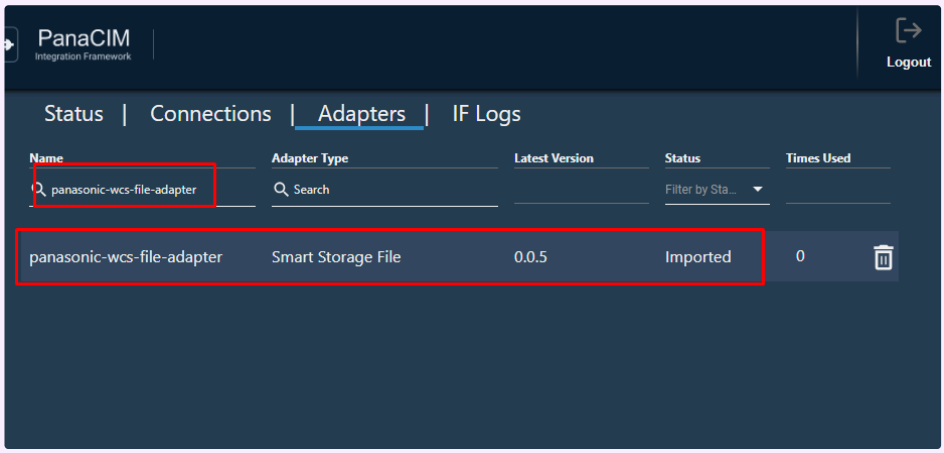


6. Click **Upload**.



The message appears when the adapter zip file is successfully uploaded.

Note: Ensure that the uploaded adapters are listed in the Adapters tab and are active.



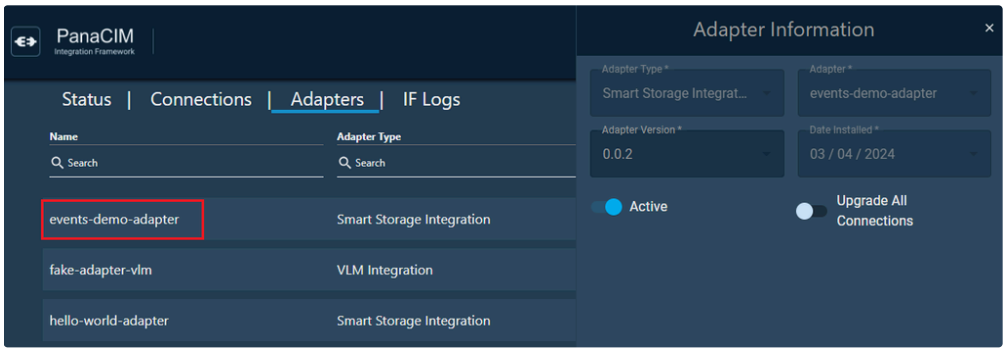
Configuring IF Adapters

Once the adapter has been successfully imported, you can manage its versions. By selecting an adapter from the list, you can access its information and navigate through its versions using the adapter version dropdown menu.

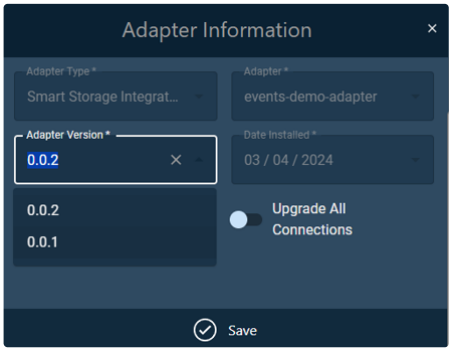
Note: Adapters are important as they are related to many functions and microservices in SH. Please ensure that you understand what you are doing before making changes to the adapter.

Follow these steps to configure IF adapters:

1. From the IF Manager interface, click on the **Adapters** tab.
2. Click on the adapter from the list to edit its version.
   The Adapter Information window opens.

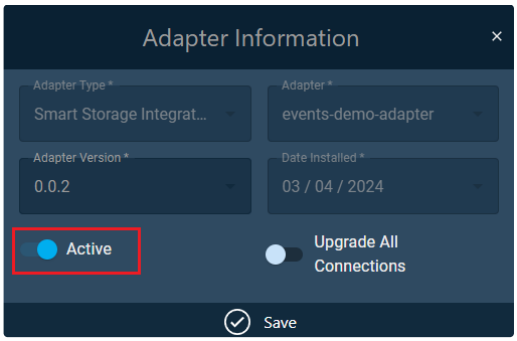


3. Select the **Adapter Version** from the drop-down list.



Note: Integration Framework will only import adapter versions greater than the maximum version of the given adapter currently in the system.

4. Click the **Active** toggle switch to enable or disable the adapter version.



Note:

- If a particular version of the adapter is not in use, it can be deactivated using the "Active" toggle switch. The deactivated version will not appear in the "Adapter Version" field when creating a new connection. This ensures that only active versions are visible as options while creating new connection. Disabling an adapter version may cause errors for connections that use that adapter version.
- If you want to upgrade all connections associated with the adapter, toggle the **Upgrade All Connections** switch.

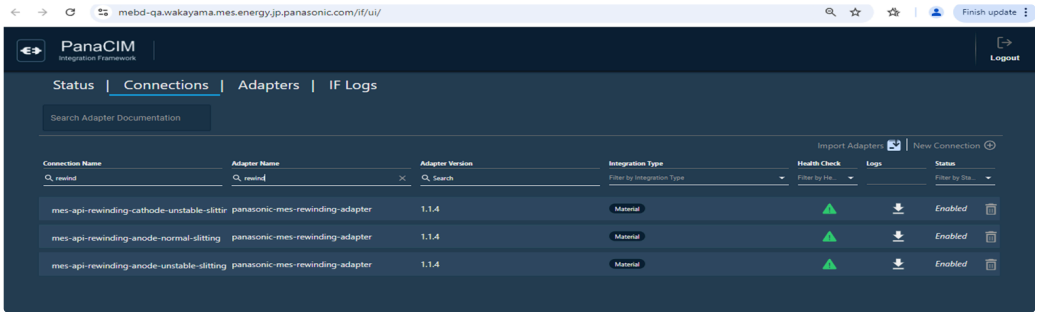
5. Click **Save**.

Creating/Configuring Connections

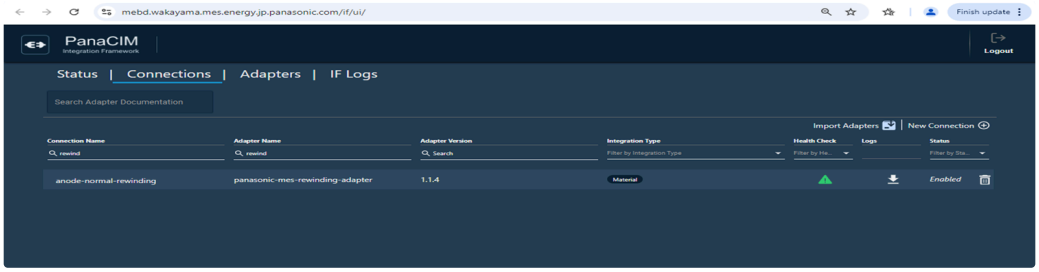
Think of a Connection as an instance of an Adapter. It leverages most of the default configurations from the Adapter template but allows for specific adjustments based on the environment (Test, DEV, QA, PROD).

Once the adapter is successfully imported, you will be able to set up a new connection in the “Connections” tab for the available adapter versions by entering all the parameters previously configured in the metadata .JSON file.

There are three connections in the MEBD QA environment.

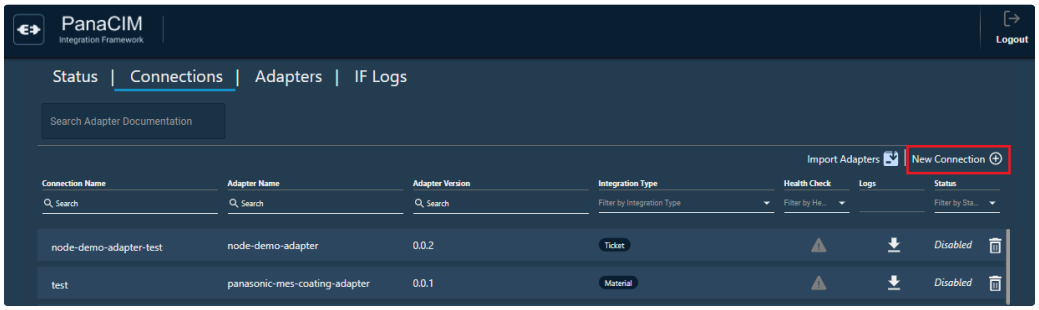


There is only one connection in the MEBD Prod environment.

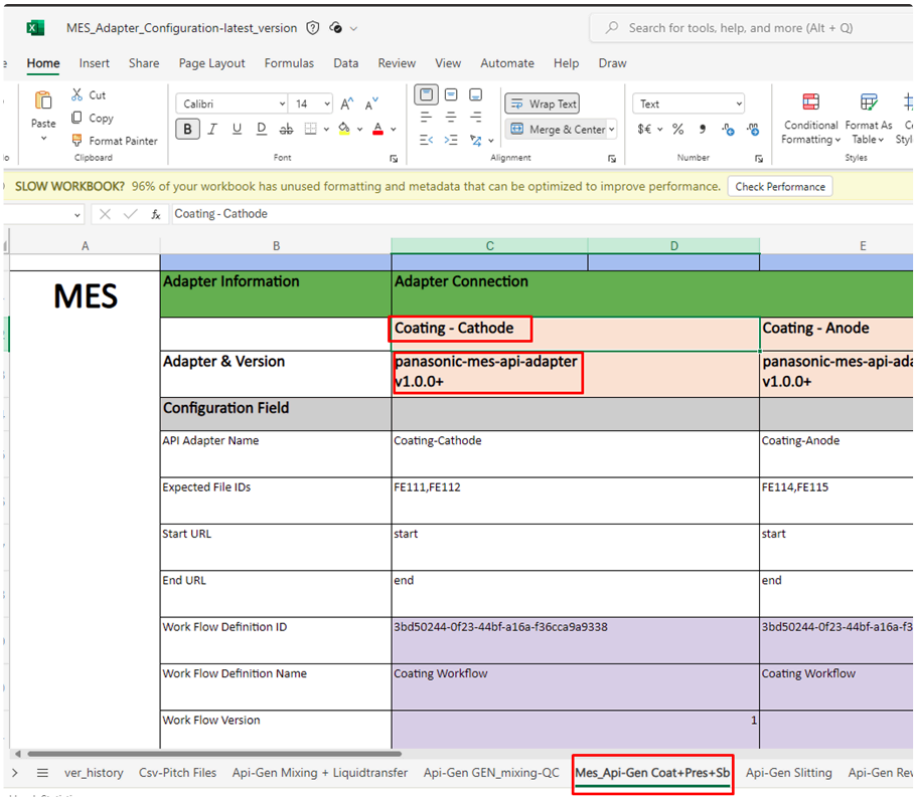


Follow these steps to create/configure connections:

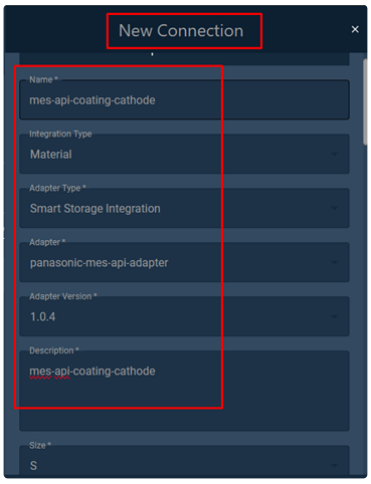
1. From the IF Manager interface, click on the **Connections** tab.
2. Click the **New Connection**.



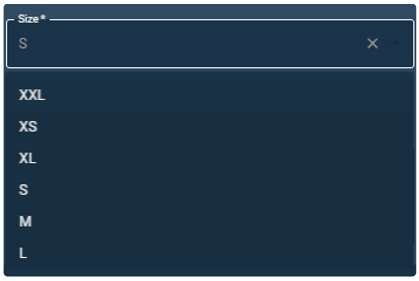
3. Enter the connection name and other information as listed in the document.
   For example, the configuration will follow the setup of "Coating - Cathode."



For description of the adapter fields, refer IF Adapter | Description of the Adapter Fields.



4. Select the following parameters that are previously configured in the metadata .JSON file:
   - Integration Type
   - Adapter Type
   - Adapter
   - Adapter Version
5. Enter the **Description** for the new connection.
6. Modify the **Size** from the dropdown list.



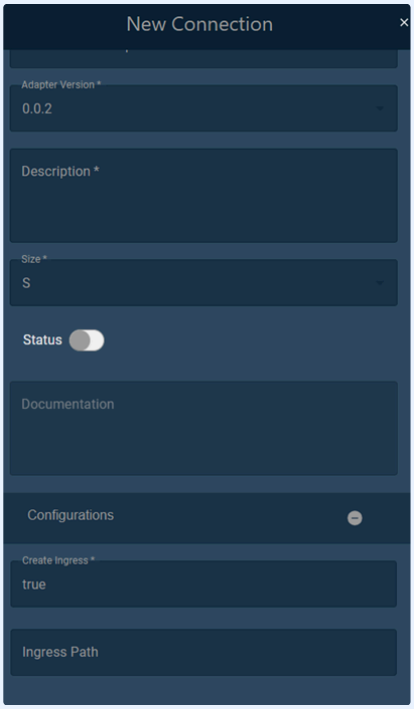
The size parameter is linked to the "RequestedResourceSize" that was previously set in the default adapter metadata. The T-shirt size that is associated refers to a predetermined amount of CPU and memory present in the database, and it will be used as a parameter for the requests and limits in the deployment.

Below is a description of the meaning of each size and the corresponding CPU/memory requests and limits for each one:

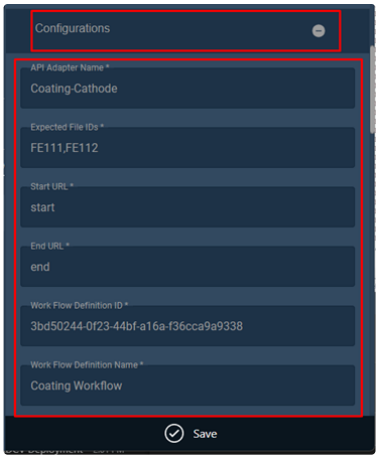
Size	Description	CPU Request	CPU Limit	Memory Request	Memory Limit
XS	Extra Small	50m	75m	100Mi	150Mi
S	Small	100m	150m	150Mi	225Mi
M	Medium	150m	225m	300Mi	450Mi
L	Large	300m	450m	600Mi	900Mi
XL	Extra Large	600m	900m	1200Mi	1800Mi
XXL	Double Extra Large	1200m	1800m	2400Mi	3600Mi

7. Enable the **Status** toggle of the connection.

If the Status is set as enabled, the connection will be successfully deployed in the cluster with the provided configurations. Alternatively, if it is disabled, resources such as services and ingress will be removed, and the deployment will be scaled down to zero. Certain deployment configurations can only be changed when the connection is disabled.

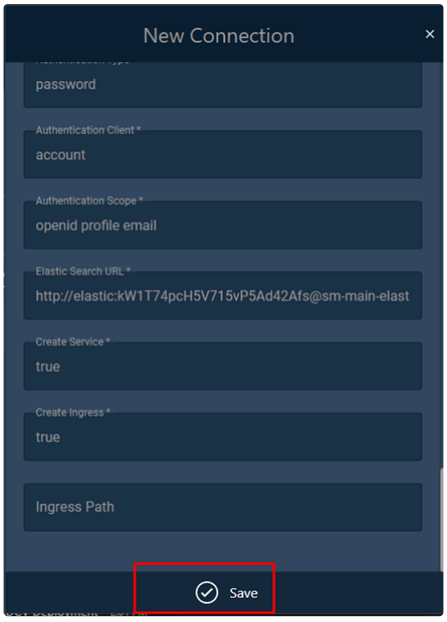


8. Click the + icon to expand the “Configuration” and check values in each field.



Note: Ensure that the values in each field match the values in the configuration file. For configuration values, refer MES Adapter Configuration. The configuration fields are represented in different colors; refer to the "Significance of color in the table".

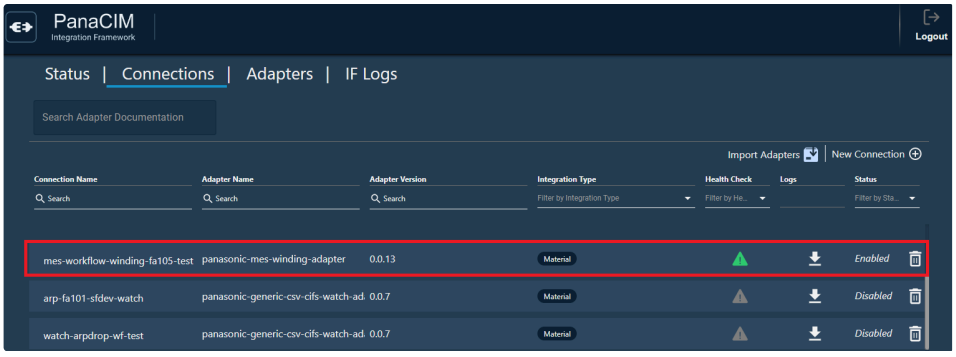
9. Click **Save**.



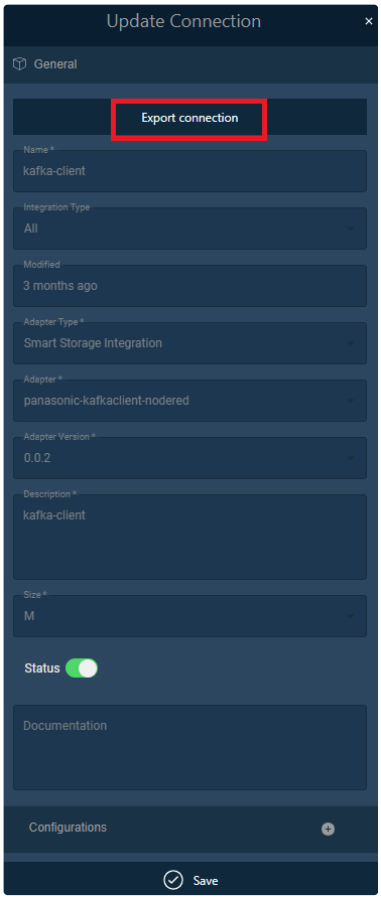
Exporting IF Connections

Follow these steps to export IF connections:

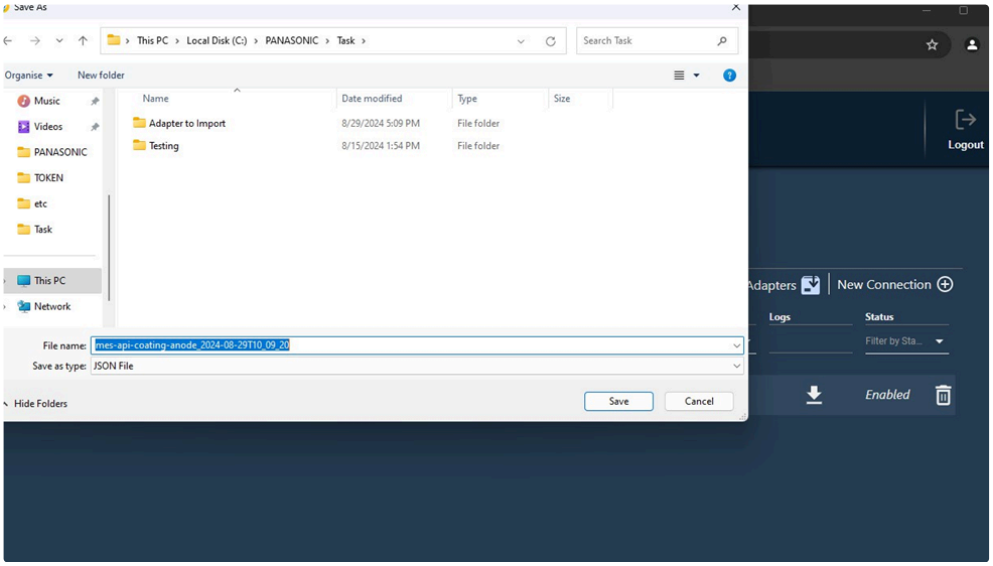
1. From the IF Manager interface, click on the **Connections** tab.
2. Select the connection you want to export from the list.
   The Update Connection window opens.



3. Click **Export connection**.



4. Select the location on your local PC to export the JSON file.



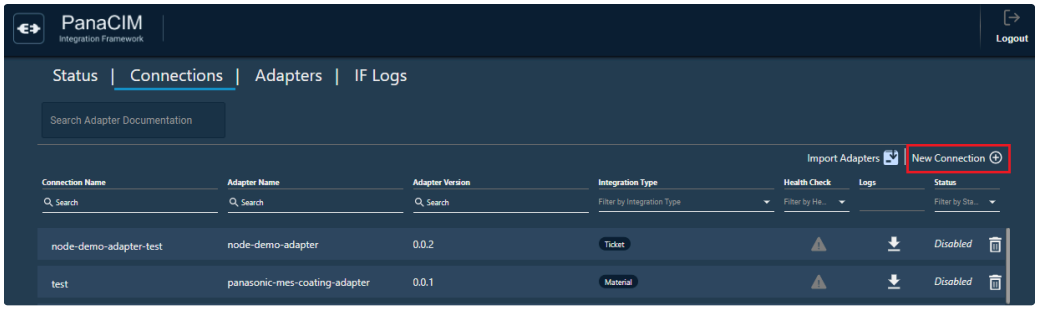
Note: Ensure the exported file is compatible with the version of the IF Manager in the target environment.

7. Click **Save.**

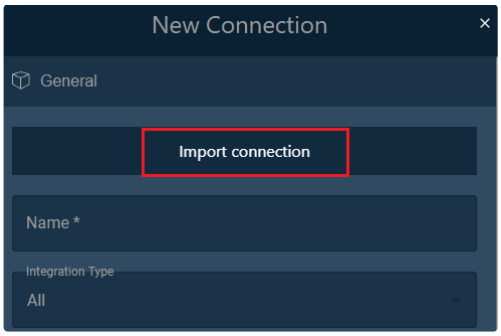
Importing IF Connections

Follow these steps to import IF connections:

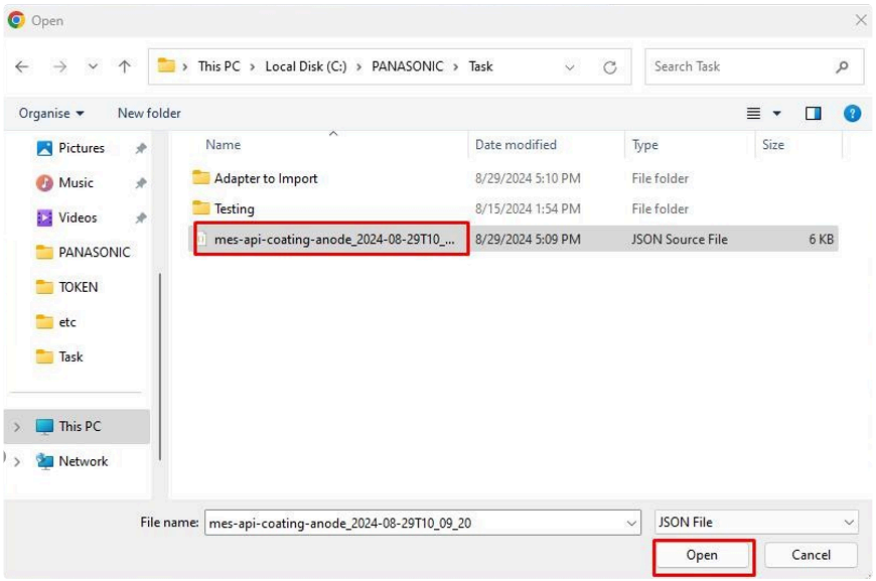
1. From the IF Manager interface, click on the **Connections** tab.
2. Click the **New Connection**.



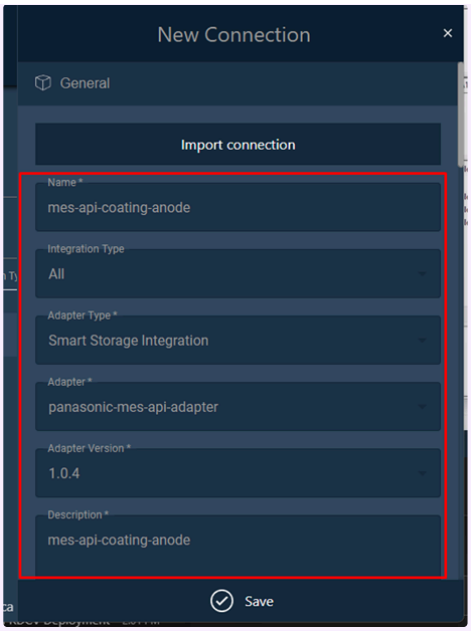
3. Click **Import connection**.



4. Select the downloaded JSON connection file from your local drive and click **Open**.
   The details of the imported connection file are displayed.



Note: Ensure that the values in each field match those in the downloaded JSON connection file.

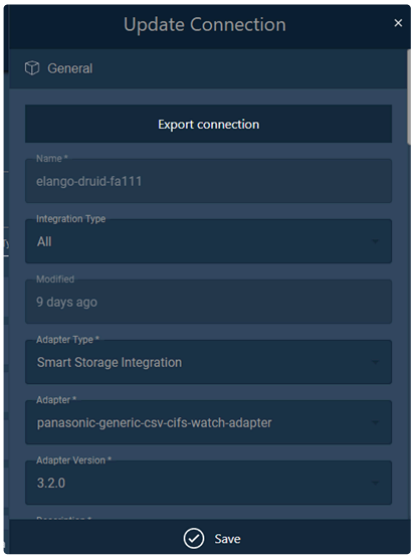


5. Click **Save**.

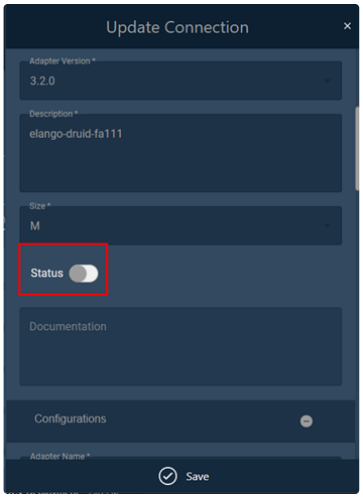
Editing Connections

Follow these steps to edit connections:

1. From the IF Manager interface, click on the **Connections** tab.
2. Click on the connection that you want to edit.
   The Update Connection window opens.

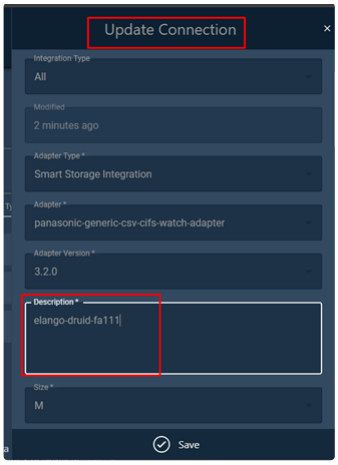


3. Disable the connection by clicking the “Status” toggle switch.



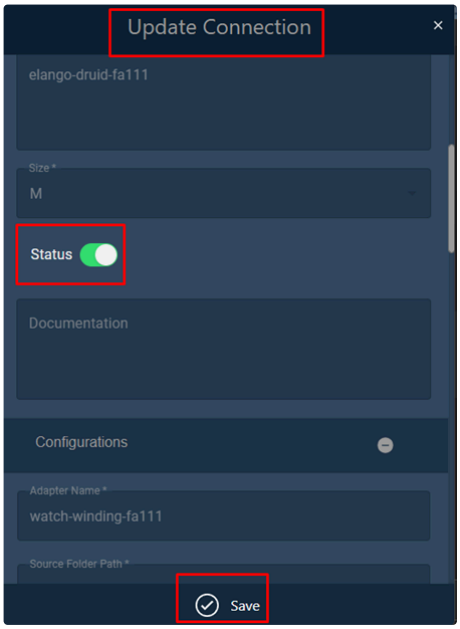
Note: Only the disabled connection can be edited.

4. Click **Save**.
5. Select the connection again to edit the values in the fields.



Note: The connection is crucial as it is related to many functions and microservices in SH. Ensure that you fully understand the implications before making any changes to the connection settings.

6. After editing, enable the connection by clicking the “Status” toggle switch.



Note: Ensure that the Status toggle switch is green to enable the connection.

7. Click **Save**.

Master Data

Master Data is a crucial asset for any organization. It provides the fundamental information necessary for efficient operations and informed decision-making. Effective Master Data management is vital for ensuring data consistency, accuracy, and integrity across different systems and applications.

Importing Master Data

This procedure outlines the steps to create, test, and import master data into a production environment using a template for the Mobility Energy Business Division (MEBD). Also, it ensures that master data is thoroughly tested and validated before being deployed to the production environment, minimizing the risk of errors and ensuring data integrity.

Follow these steps to import master data for MEBD:

1. Create the master data file using the provided template.
   Ensure all necessary fields are filled correctly and consistently.
2. Import the master data file into the Quality Assurance (QA) environment for initial testing.
3. Perform thorough testing in the QA environment to verify the accuracy and functionality of the imported data.
4. Identify and document any errors or issues.
5. Based on the testing results, make necessary corrections to the template. Update the master data file to address identified errors.
6. Revert the QA environment to its original, clean state (gold version) before the initial data import.

> **Note**: Ensure the environment is free of any previously imported data.

7. Reimport the corrected and updated master data template into the QA environment.
8. Conduct another round of testing in the QA environment to verify the accuracy and functionality of the reimported data.
9. Repeat this procedure in QA until User Acceptance Testing (UAT) is passed.
10. Once UAT is successfully passed, use the validated template to load all master data into the production server.
11. Perform a spot check to ensure the data has been imported correctly and is functioning as expected.

> If this template can be used to import master data, it will save work; otherwise, the master data can be translated into the final template.

Workflows

A workflow is a series of tasks that represents a process or activity on the factory floor. It helps standardize and streamline task execution. In SmartHive, a workflow allows you to visually model business processes. You can then streamline or automate these processes by setting up triggers across accessible applications.

The workflow acts as a generic execution engine, capable of interacting with any element in the MES platform. This flexibility allows workflows to be customized for various entities, such as Material, Product, Station, and more. Additionally, they can execute custom logic as needed. Users can monitor and guide any object (such as Material, Product, Carrier, etc.) through a workflow, enabling comprehensive control and execution.

A workflow in SmartHive typically involves the following elements:

1. **Tasks:** Individual steps or actions that need to be performed during the process.
2. **Triggers:** Events or conditions that initiate the workflow. These could be actions performed by users, a time-based trigger, or the occurrence of certain data-related events.
3. **Automation:** Predefined rules or logic that govern how the tasks are executed based on triggers and other relevant factors.
4. **Notifications:** Alerts or messages that can be sent to relevant parties to keep them informed about the progress or completion of workflows.
5. **Approvals:** Steps associated with tasks that require authorization from other users.

Workflows are a fundamental component of the SmartHive ecosystem, enabling the automation and orchestration of complex processes across various systems and applications. By designing, managing, and optimizing workflows effectively, organizations can achieve significant improvements in efficiency, accuracy, and scalability, ultimately enhancing their overall operational performance. For the list of workflows, refer List of Workflows.

Licensing

Troubleshooting

The troubleshooting section covers various potential issues across different system components. It involves several steps and strategies to diagnose and resolve problems, ensuring minimal downtime and efficient issue resolution.

You can effectively troubleshoot issues within your environment by systematically addressing each area. Always start with gathering detailed information through logs and monitoring tools to pinpoint the root cause of the problem.

Environmental Issues

Troubleshooting Kafka Issues

Troubleshooting Kafka issues can involve various steps depending on the specific problem you are encountering.

Follow these steps to troubleshoot common Kafka issues:

1. Identify the problem area and check logs.
   Some of the common issues and solutions are provided in the following table:

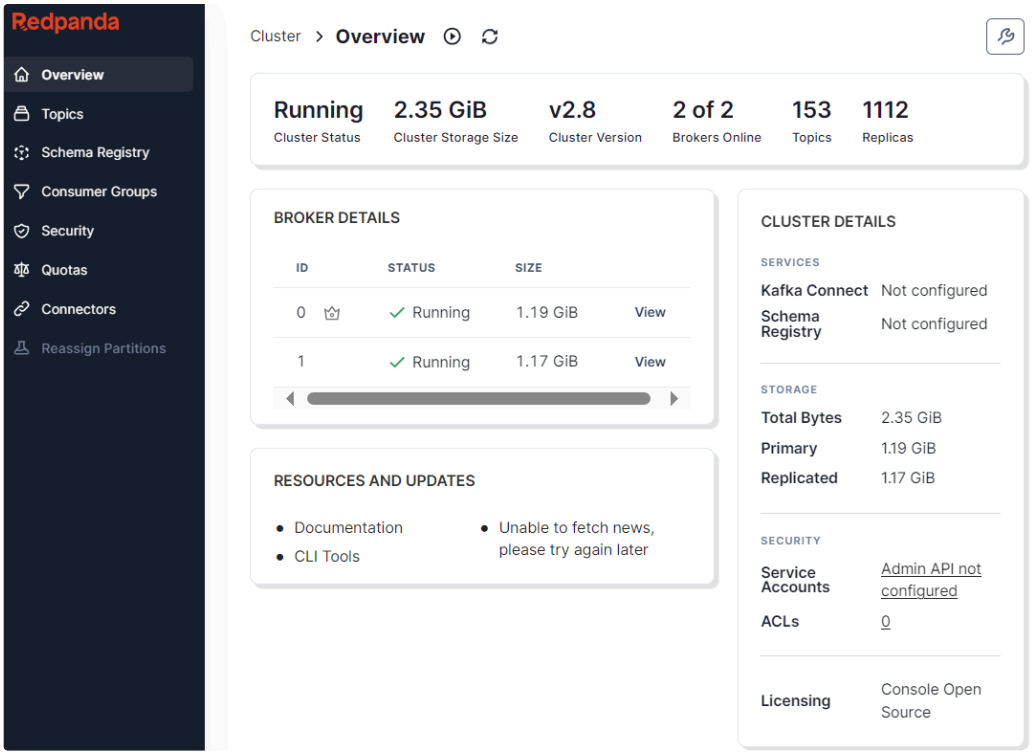
Common Issues	Solution
Producer Issues	
Message Not Sent	• Check producer logs for errors. • Ensure the broker is reachable. • Verify the topic exists and the producer has write permissions. • Check for configuration errors (e.g., `acks`, `retries`).
High Latency	• Monitor network latency. • Check broker load and resource utilization. • Ensure proper batching settings in the producer configuration.
Broker Issues	
Broker Not Starting	• Check for port conflicts. • Verify configuration files (`server.properties`). • Ensure Zookeeper is running and reachable.
Under-Replicated Partitions	• Check the health of all brokers. • Verify that brokers are not overloaded. • Ensure proper replication factor and in-sync replicas (ISR).
Consumer Issues	
Lagging Consumers	• Monitor consumer lag using Kafka tools (e.g., Kafka Manager, Burrow). • Check consumer group configuration and offsets. • Ensure consumers are balanced across partitions.
Unable to Consume Messages	• Check consumer logs for errors. • Ensure the topic exists and the consumer has read permissions.

	• Verify offset management and group ID configuration.
Zookeeper Issues	
Zookeeper Not Starting	• Check Zookeeper configuration files (`zoo.cfg`). • Ensure correct data directory and log directory paths. • Verify Zookeeper port is available.
Connectivity Issues	• Ensure network connectivity between Zookeeper and brokers. • Check for firewall rules blocking the connection. • Verify Zookeeper ensemble configuration if using multiple nodes.

2. Monitor Kafka metrics.
   - Use tools like Kafka Metrics, JMX, Prometheus, and Grafana to monitor Kafka metrics.
   - Key metrics to watch: `Message In/Out`, `Consumer Lag`, `Request Latency`, `ISR Shrinks/Expands`, `Under-Replicated Partitions`.
3. Check the configuration.
   - Verify your `server.properties`, `producer.properties`, `consumer.properties`, and `zoo.cfg` files for correctness.
   - Ensure configuration values are appropriate for your environment and workload.
4. Use Kafka tools for the following:
   - kafka-topics.sh: For managing topics.
   - kafka-consumer-groups.sh: For managing consumer groups and offsets.
   - kafka-configs.sh: For managing broker, topic, and client configurations.
   - kafka-broker-api-versions.sh: To check broker API versions.
5. Ensure you are running the latest stable version of Kafka. Apply any patches or updates that might address known issues.

Troubleshooting Kafka Issues in Redpanda

Redpanda is a Kafka-compatible streaming data platform, and most Kafka troubleshooting techniques will apply. To access the Redpanda, click https://dev-mes.us.panasonic.com/redpanda/overview.



When troubleshooting Kafka issues in Redpanda, the process can involve several steps. Here are some common issues and steps for troubleshooting:

1. **Connection Issues**
   - **Problem:** Producers or consumers cannot connect to the Redpanda cluster.
   - **Solution:**
     - Verify the `bootstrap.servers` configuration.
     - Ensure that the Redpanda nodes are running and accessible.
     - Check for network issues, such as firewalls blocking access.
     - Use tools like `telnet` or `nc` to test connectivity to Redpanda brokers.
2. **Authentication and Authorization**
   - **Problem:** Authentication failures when connecting to Redpanda.
   - **Solution:**
     - Verify that the correct credentials (username and password) are being used.
     - Ensure that the client is configured to use the correct SASL mechanism (e.g., `SASL_PLAINTEXT` or `SASL_SSL`).
     - Check Redpanda's ACL (Access Control List) configurations.
3. **Topic Configuration Issues**
   - **Problem:** Issues with creating or accessing topics.
   - **Solution:**
     - Verify topic creation settings (number of partitions, replication factor).
     - Ensure that the topic exists by listing topics using `rpk topic list`.
     - Check topic-level configurations such as retention settings and cleanup policies.
4. **Broker Configuration Issues**
   - **Problem:** Misconfiguration of Redpanda brokers.
   - **Solution:**
     - Verify Redpanda broker configurations in the `redpanda.yaml` file.
     - Ensure that all brokers in the cluster are configured correctly and consistently.
     - Check for any broker logs that might indicate configuration issues.

5. **Performance Issues**
   - **Problem:** High latency, low throughput, or frequent timeouts.
   - **Solution:**
     - Monitor Redpanda metrics using `rpk` or other monitoring tools.
     - Optimize producer and consumer configurations (e.g., batch size, linger.ms, fetch.min.bytes).
     - Check for any resource bottlenecks (CPU, memory, disk I/O) on the brokers.
     - Ensure that the system has adequate hardware resources.
6. **Message Serialization Issues**
   - **Problem:** Errors related to message serialization.
   - **Solution:**
     - Ensure that the producer and consumer are using compatible serializers (e.g., JSON, Avro, Protobuf).
     - Verify the schema used for serialization and deserialization.
     - Check for any exceptions or errors in the client application logs.

Tools and Commands for Troubleshooting

1. **Redpanda CLI (rpk)**
   - List Topics: `rpk topic list`
   - Describe Topic: `rpk topic describe <topic-name>`
   - View Broker Configuration: `rpk config export`
2. **Kafka CLI Tools**
   - List Topics: `kafka-topics.sh --list --bootstrap-server <redpanda-broker>:9092`
   - Describe Topic: `kafka-topics.sh --describe --topic <topic-name> --bootstrap-server <redpanda-broker>:9092`
   - Producer Performance Test: `kafka-producer-perf-test.sh --topic <topic-name> --num-records 100000 --record-size 100 --throughput -1 --producer-props bootstrap.servers=<redpanda-broker>:9092`
   - Consumer Performance Test: `kafka-consumer-perf-test.sh --topic <topic-name> --messages 100000 --bootstrap-server <redpand`

Troubleshooting Rest API and Kafka Ingestion

Troubleshooting REST API and Kafka ingestion involves identifying and resolving issues that may arise during data ingestion from a REST API into Kafka.

Follow these troubleshooting steps and solutions to identify and resolve issues related to REST API and Kafka ingestion:

Troubleshooting REST API Issues

1. **API Endpoint Accessibility**
   - **Problem:** The REST API endpoint is not reachable.
   - **Solution:** Verify the endpoint URL, check network connectivity, and ensure there are no firewall issues blocking access. Use tools like `curl` or `Postman` to test the endpoint.
2. **Authentication and Authorization**
   - **Problem:** Authentication errors (e.g., `401 Unauthorized`).
   - **Solution:** Ensure correct API keys, tokens, or credentials are being used. Check for any required headers and ensure they are included in the request.
3. **Rate Limiting**
   - **Problem:** API rate limits are being exceeded.
   - **Solution:** Implement rate limiting and retry mechanisms in your client code. Respect the API's rate limits as specified in the documentation.
4. **Data Format Issues**

- **Problem:** Data format issues such as incorrect JSON structure.
- **Solution:** Validate the JSON payloads being sent to the API. Ensure they conform to the expected schema.

5. **Error Handling**
   - **Problem:** Unhandled API errors causing ingestion failures.
   - **Solution:** Implement robust error handling and logging. Handle different HTTP status codes appropriately.

Troubleshooting Kafka Ingestion Issues

1. **Producer Configuration**
   - **Problem:** Kafka producer configuration issues leading to connection failures.
   - **Solution:** Verify producer configuration settings such as `bootstrap.servers`, `acks`, and `retries`. Ensure the producer can reach the Kafka brokers.
2. **Broker Availability**
   - **Problem:** Kafka brokers are not available or are down.
   - **Solution:** Check the status of Kafka brokers. Ensure they are running and accessible. Use tools like `kafka-topics.sh` to check the status of topics and partitions.
3. **Topic Configuration**
   - **Problem:** Issues with the Kafka topic configuration.
   - **Solution:** Verify the topic exists and has the correct number of partitions and replication factor. Ensure the producer is publishing to the correct topic.
4. **Message Serialization**
   - **Problem:** Serialization errors when producing messages.
   - **Solution:** Ensure the producer and consumer are using compatible serializers (e.g., JSON, Avro). Verify the message format being sent to Kafka.
5. **Error Handling in Producers**
   - **Problem:** Unhandled errors in Kafka producers.
   - **Solution:** Implement error handling and retries in the producer code. Log errors and monitor for any recurring issues.
6. **Network Issues**
   - **Problem:** Network connectivity issues between producers and Kafka brokers.
   - **Solution:** Check network connectivity and firewall settings. Ensure there are no network partitions or high latency affecting communication.

Troubleshooting REST API Issues

When troubleshooting REST API issues, follow a systematic approach to identify and resolve problems.

Follow these troubleshooting steps and solutions to identify and resolve REST API issues:

1. Check the Basics
   a. Verify the Endpoint URL.
      - **Issue:** Incorrect URL.
      - **Solution:** Ensure the URL is correct, including the protocol (http/https), domain, and path.
   b. Check HTTP method.
      - **Issue:** Using the wrong HTTP method (GET, POST, PUT, DELETE, etc.).
      - **Solution:** Verify that you are using the correct method as specified by the API documentation.
   c. Validate request parameters.
      - **Issue:** Missing or incorrect parameters.
      - **Solution:** Check the API documentation for required query parameters, headers, and body data. Ensure all required parameters are included and correctly formatted.
2. Authentication and Authorization

a. Verify authentication details.
   - **Issue:** Missing or incorrect authentication.
   - **Solution:** Ensure that API keys, tokens, or credentials are correctly included in the request headers or parameters.

b. Check authorization.
   - **Issue:** Insufficient permissions.
   - **Solution:** Verify that the user or application has the necessary permissions to access the requested resource.

3. Inspect Request and Response

a. Inspect request and response using API testing tools.
   - **Tools:** Postman, curl, Insomnia.
   - **Action:** Use these tools to manually test API endpoints and inspect request/response details.

b. Check response status codes.
   - **Issue:** Non-200 status codes.
   - **Solution:**
     - `400 Bad Request`: Check for malformed request syntax or invalid parameters.
     - `401 Unauthorized`: Check authentication credentials.
     - `403 Forbidden`: Check permissions.
     - `404 Not Found`: Verify the endpoint URL and resource existence.
     - `500 Internal Server Error`: Check server logs for errors.

4. Debugging Data Issues

a. Inspect request payload.
   - **Issue:** Incorrect or malformed request body.
   - **Solution:** Verify the JSON or XML structure, field names, and data types.

b. Check response payload.
   - **Issue:** Unexpected or missing data in response.
   - **Solution:** Inspect the response body for error messages or incorrect data. Ensure the server is returning the expected format and data.

5. Network and Connectivity

a. Check network connectivity.
   - **Issue:** Network issues.
   - **Solution:** Ensure the client can reach the server. Use tools like `ping`, `traceroute`, or browser developer tools to check network connectivity.

b. Inspect firewall and proxy settings.
   - **Issue:** Network firewall or proxy blocking requests.
   - **Solution:** Verify that firewalls or proxies are not blocking the requests. Ensure that ports used by the API are open.

6. Server-Side Issues

a. Check server logs.
   - **Issue:** Server errors.
   - **Solution:** Inspect server logs for error messages or stack traces that indicate the root cause of the problem.

b. Monitor server performance.
   - **Issue:** Server overload or performance issues.
   - **Solution:** Monitor server resources (CPU, memory, disk usage) and ensure the server can handle the request load.

7. Client-Side Debugging

a. Use browser developer tools
   - **Tools:** Chrome DevTools, Firefox Developer Tools.

- **Action:** Inspect network requests and responses, console errors, and performance metrics.

b. Debug client code

- **Issue:** Client-side errors.
- **Solution:** Debug client application code to ensure correct handling of API requests and responses.

8. API Documentation and Updates

a. Review API documentation.

- **Issue:** Misunderstanding API usage.
- **Solution:** Thoroughly review the API documentation for correct usage, parameters, and examples.

b. Check for API updates.

- **Issue:** API changes or deprecation.
- **Solution:** Verify if the API has been updated or deprecated. Ensure the client code is compatible with the current API version.

Software Connections

Software connections are essential for enabling seamless data exchange, functionality integration, and coordinated operations across different platforms and devices.

Debugging communication issues between software systems like PPE (Panasonic Production Engineering), WCS (Warehouse Control System), and EdgePC (Edge Computing System) involves a thorough and systematic approach to identify where the communication breakdowns are occurring and resolve them.

Troubleshooting Communication Issues

Follow these steps to debug and resolve communication issues:

> **Note**: The SmartHive mobile app communicates only with PPE, not with other systems. It will display the message "Error: Internal server error" when the PPE is down or if there's an unhandled error on the PPE side.
>
> In general, you may encounter a "network request failed" error when the mobile device is not connected to the internet, intranet, or Wi-Fi.

1. Understand the data flow and interaction between PPE, WCS, and EdgePC.
2. Document the protocols, endpoints, and services involved.
3. Use `ping` and `traceroute` (or `tracert` on Windows) to check the basic connectivity between systems.
   `ping <target-IP> traceroute <target-IP>`
4. Use tools like `netcat`, `telnet`, or `nc` to test connectivity on specific ports.
   `nc -zv <target-IP> <port>`
5. Ensure that IP addresses and ports are correctly configured on all systems.
6. Check firewall settings to ensure that communication ports are open.
7. If applicable, ensure Network Address Translation (NAT) and Virtual Private Network (VPN) settings are correctly configured.
8. Check logs on all involved systems for error messages or warnings.
   - PPE logs
   - WCS logs
   - EdgePC logs
9. If available, use centralized logging tools like ELK Stack (Elasticsearch, Logstash, Kibana) or Splunk.
10. Verify that the correct network protocols (HTTP, MQTT, AMQP, etc.) are being used.
11. Use Wireshark to capture and analyze network traffic between systems to identify anomalies or errors.
12. Test API endpoints individually using tools like Postman or cURL to ensure they are responding correctly.
    `curl -X GET http://<endpoint>`
13. Check the status of services (e.g., REST APIs, MQTT brokers) to ensure they are running and reachable.
14. If using message queues (e.g., RabbitMQ, Kafka), verify that messages are being published and consumed correctly.
15. Ensure that message brokers are running without issues.
16. Ensure all systems have synchronized clocks (using NTP) to avoid timing issues.
17. Verify if there is appropriate retry logic in place for transient communication failures.
18. Check SSL/TLS certificates and encryption settings for secure communication.
19. Ensure that authentication tokens or API keys are correctly configured and valid.
20. Ensure that all software components are compatible and up to date.
21. Verify that all required dependencies are correctly installed and compatible.

22. If using containerized environments, use `kubectl` or `docker` commands to inspect pod and container statuses.
    `kubectl logs <pod-name> docker logs <container-id>`
23. Use edge-specific diagnostic tools provided by the hardware or software vendor.
24. Work with network engineers, system administrators, and developers to gather different perspectives and expertise.
25. Maintain detailed documentation of configurations, error messages, and troubleshooting steps to facilitate resolution and future reference.
26. Replicate the issue in a controlled test environment if possible to isolate and debug.
27. Perform load testing to see how systems behave under different loads and identify potential bottlenecks.

Checking Connection Between Databases

Checking connections between databases, whether they are in the cloud or on-premises, involves a series of steps to verify network connectivity, database accessibility, and proper configuration.

> **Note**: The mobile app does not connect directly to the database. Instead, it is connected to microservices, just like the web app. Users must specify the domain URL of the API in the settings screen of the mobile app. This allows to identify the environment connected to the mobile app using the settings screen.

Follow these steps to check and troubleshoot connection between databases:

1. Use the `ping` command to check if the database server is reachable.
   `ping <database-server-ip-or-hostname>`
2. Use `traceroute` (or `tracert` on Windows) to determine the network path to the database server and identify any network bottlenecks.

   `traceroute <database-server-ip-or-hostname>`
3. Use `telnet` or `nc` to test if the database server's port is open and accessible.
   `telnet <database-server-ip-or-hostname> <port> # or nc -zv <database-server-ip-or-hostname> <port>`
4. Use database-specific client tools (e.g., MySQL Workbench, pgAdmin, SQL Server Management Studio) to connect to the database. These tools often provide more detailed error messages.
5. Use command-line clients to connect to the database.
   - MySQL/MariaDB:
     `mysql -h <database-server-ip-or-hostname> -u <username> -p`
   - PostgreSQL:
     `psql -h <database-server-ip-or-hostname> -U <username> -d <database-name>`
   - SQL Server:

     `sqlcmd -S <database-server-ip-or-hostname> -U <username> -P <password>`
6. Check the application’s database connection configuration files for correct settings (hostname, port, database name, username, password).
7. Review application logs for any database connection errors.
8. Ensure environment variables related to database connections are correctly set.
9. Ensure that the database server's firewall allows inbound connections on the database port.
10. For cloud databases, verify that security groups or network ACLs allow traffic from the client to the database port.
11. Ensure the database user has the necessary privileges to connect and perform required operations.
12. Verify that the database user can connect from the client's IP address (some databases restrict access by IP).
13. Ensure the database server is configured to listen on the correct IP address and port.
14. Check if the database server has any connection limits that might be preventing new connections.
15. Use a simple Python script to test database connectivity.
    `import mysql.connector try: connection = mysql.connector.connect( host='<database-server-ip-or-hostname>', user='<username>', password='<password>', database='<database-name>' ) if connection.is_connected():`

`print("Connection successful") except Exception as e: print(f"Error: {e}") finally: if connection.is_connected(): connection.close()`

16. Use a shell script to test connectivity.
    `#!/bin/bash mysql -h <database-server-ip-or-hostname> -u <username> -p<password> -e "SHOW DATABASES;"`
17. Use the AWS RDS console or CLI to check the status and endpoint of the RDS instance.
    `aws rds describe-db-instances --db-instance-identifier <db-instance-id>`
18. Use the Azure portal or CLI to check the status and connection information of the Azure SQL Database.
    `az sql db show --resource-group <resource-group> --server <server-name> --name <database-name>`
19. Use the GCP console or CLI to check the status and endpoint of the Cloud SQL instance.
    `gcloud sql instances describe <instance-name>`
20. Continuously monitor logs for any connectivity issues.
21. Keep a record of configurations, changes made during troubleshooting, and any resolved issues for future reference.

Troubleshooting Software Drivers

Debugging software drivers for hardware devices such as printers, scanners, and RFID gates involves several steps to identify and resolve issues that might be causing the hardware to malfunction or not communicate properly with the system.

> **Note**: The SmartHive mobile app does not need drivers to be connected to the scanner.

Follow these steps to identify and resolve issues with software drivers:

1. Verify that the device is properly connected to the computer or network.
2. Turn off the device, wait a few seconds, and then turn it back on.
3. On Windows, open the Device Manager to see if the device is recognized and if there are any error messages.
4. Use the operating system’s automatic update feature to check and install the latest drivers.
5. Download and install the latest drivers from the device manufacturer's website.
6. Check if there is a firmware update available for the device and install it if applicable.
7. Go to Device Manager, right-click the device, and select "Uninstall Device."

> **Note**: Make sure to check the option to delete the driver software for this device if available.

8. Restart the computer after uninstalling the driver.
9. Install the latest driver from the manufacturer’s website.
10. Look for yellow exclamation marks or red **X** in Device Manager that indicate conflicts or errors.
11. Ensure that the device is not conflicting with another device in terms of I/O addresses, IRQs, or DMA channels.

> **Note**: Temporarily disable other connected devices to see if they are causing conflicts.

12. Use any diagnostic tools provided by the device manufacturer.
13. Run the built-in Windows troubleshooter for hardware and devices.
14. Use third-party diagnostic tools like Driver Booster or DriverPack Solution to detect and fix driver issues.
15. On Windows, open the Event Viewer to check for system and application logs related to driver or hardware issues.
16. Check the specific log files generated by the device’s driver or management software.
17. Access the driver settings through Device Manager or the device’s control panel to ensure proper configuration.
18. Check any accompanying software for the device (e.g., printer software) to ensure settings are correct.
19. **Test on Another System**
    - Connect the device to another computer to determine if the issue is with the device or the original system.
    - If possible, test the device on a different operating system to rule out OS-specific issues.

20. **Contact Support**
    - Contact the device manufacturer’s technical support for assistance.
    - Search for similar issues on user forums and communities to find potential solutions.
    - If the issue persists, consider seeking help from a professional technician.

Example: Debugging a Printer Driver

1. Ensure the printer is connected and powered on.
2. Check the connection cable or Wi-Fi settings.
3. Visit the printer manufacturer’s website to download and install the latest driver.
4. Uninstall the existing printer driver via Device Manager.
5. Reboot and reinstall the driver.
6. Check no other USB devices are causing conflicts.
7. Run the printer’s built-in diagnostic tool or the Windows printer troubleshooter.
8. Review logs in the Event Viewer for error messages related to the printer.
9. Verify printer settings in the control panel and driver software.
10. Connect the printer to a different computer to check if it works.
11. Contact printer support or visit forums if the problem persists.

Hardware Connections

Debugging hardware connections for devices such as tablets, scanners, monitors, and Wi-Fi routers involves a systematic approach to identify and resolve issues.

Troubleshooting Hardware Connections

Follow these steps to debug and resolve the hardware connection issues:

1. **Initial Checks**
   - Ensure all cables are securely connected. For wireless devices, check the connection status.
   - Turn off the device, wait a few seconds, and then turn it back on.
   - Ensure the device is receiving power. Check for any power indicator lights.
2. **Device Manager (Windows)**
   - Open Device Manager (Press `Win + X`, then select Device Manager).
   - Check for any yellow exclamation marks or red crosses next to the device.
3. **Update Drivers/Firmware**
   - Download the latest drivers/firmware from the manufacturer’s website or use the operating system’s built-in update feature to check and install updates.
   - Some devices require firmware updates which can be downloaded from the manufacturer's support page.
4. **Check Settings and Configuration**
   - Ensure the device is properly configured in the system settings.
   - Access the device’s control panel or configuration tool to ensure settings are correct.
5. **Test on Another System**
   - Connect the hardware to a different computer or device to rule out issues with the original system.
   - If possible, test the hardware on a different operating system to determine if the issue is OS-specific.
6. **Logs and Diagnostics**
   - Press `Win + X`, and select Event Viewer to check for any system logs related to the device.
   - Use any diagnostic tools provided by the device manufacturer.

Troubleshooting Specific Devices

Tablets

Common Issues and Solutions

- **Wi-Fi Connectivity Issues**:
  - **Solution**:
    i. Ensure the tablet’s Wi-Fi is turned on and try connecting to the network.
    ii. Restart the router and the tablet.
    iii. Forget the network on the tablet and reconnect.
    iv. Check for software updates on the tablet and install them.
    v. Reset network settings on the tablet (Settings > System > Reset options > Reset Wi-Fi, mobile & Bluetooth).
- **USB Connection Problems**:
  - **Solution**:
    i. Ensure the USB cable is not damaged and is properly connected.
    ii. Try a different USB port on the computer.
    iii. Check the tablet’s USB connection settings and ensure it is set to the correct mode (e.g., File Transfer).

iv. Install or update the necessary drivers on the computer.

Barcode Scanners

Common Issues and Solutions

- **Scanner Not Detected**:
  - **Solution**:
    i. Ensure that all cables are securely connected.
    ii. If using a USB scanner, try a different USB port.
    iii. For wireless scanners, ensure that the receiver is properly connected to the computer and that the scanner is charged.
- **Scanner Configuration Issues**:
  - **Solution**:
    i. Refer to the user manual to reset the scanner to factory settings.
    ii. Scan the appropriate programming barcode from the manual to reconfigure the scanner for the correct interface (USB, Serial, Keyboard Wedge, etc.).
- **Scanning Issues**:
  - **Solution**:
    i. Use a soft, lint-free cloth to gently clean the scanner lens. Avoid using abrasive cleaners or solvents.
    ii. Ensure there are no physical obstructions between the scanner and the barcode. Try scanning a different barcode to see if the issue persists.
    iii. Ensure that the barcode is printed clearly and not damaged or smudged.
    iv. Adjust the distance between the scanner and the barcode according to the scanner's specifications.
    v. Hold the scanner at a proper angle to ensure the entire barcode is within the scanner's reading area.
- **Scanner Communication Issues**:
  - **Solution**:
    i. Ensure that the scanner is configured to the appropriate interface mode (USB HID, USB COM, etc.). Refer to the scanner manual to set the correct interface mode by scanning the corresponding configuration barcodes.
    ii. Ensure that the necessary drivers are installed. Check the manufacturer's website for the latest drivers. Verify that the scanner is correctly configured in the software application you are using.
    iii. Check the manufacturer's website for firmware updates. Follow the instructions provided by the manufacturer to update the firmware.
    iv. Test the scanner on a different computer or device to determine if the issue is with the scanner or the computer.

Monitors

Common Issues and Solutions

- **No Display/Signal**:
  - **Solution**:
    i. Ensure the monitor is powered on and the power cable is connected properly.
    ii. Check the display cable (HDMI, VGA, DVI, DisplayPort) and ensure it is securely connected to both the monitor and the computer.
    iii. Select the correct input source on the monitor.
    iv. Try using a different display cable or port.
    v. Connect the monitor to another computer to determine if the issue is with the monitor or the computer.
- **Resolution or Display Issues**:
  - **Solution**:
    i. Adjust the display resolution settings on the computer to match the monitor's native resolution.

ii. Update the graphics drivers on the computer.
iii. Check for any physical damage to the display cable or ports.
iv. Reset the monitor to its factory settings.

Wi-Fi Routers

Common Issues and Solutions

- **No Internet Connection**:
  - **Solution**:
    i. Ensure the router is properly connected to the modem and powered on.
    ii. Restart both the modem and the router.
    iii. Check the ISP (Internet Service Provider) status to see if there are any outages.
    iv. Access the router's configuration page (usually by entering the router's IP address in a web browser) and check the internet connection settings.
    v. Reset the router to factory settings and reconfigure it.
- **Weak Wi-Fi Signal**:
  - **Solution**:
    i. Position the router in a central location, away from obstructions and interference.
    ii. Update the router's firmware to the latest version.
    iii. Change the Wi-Fi channel to a less congested one.
    iv. Use Wi-Fi range extenders or mesh network systems to improve coverage.
    v. Reduce the number of devices connected to the Wi-Fi network.

Diagnostic Tools and Commands

- **Device Manager**: `devmgmt.msc`
  Check for any devices with warning icons and update or reinstall drivers.
- **Network Troubleshooter**: `msdt.exe /id NetworkDiagnosticsNetworkAdapter`
- **Event Viewer**: `eventvwr.msc`
  Check for any system or application logs that might indicate hardware issues.

Infrastructure

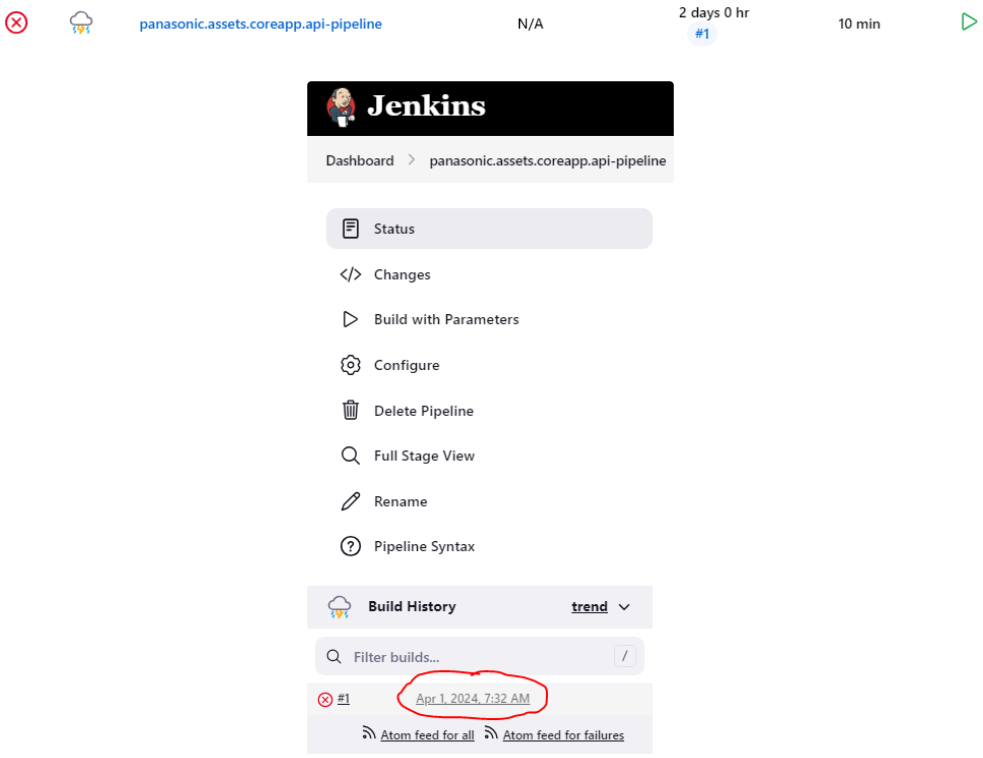
Troubleshooting Steps for Failed Deployments in Infrastructure Applications

When deployments fail in infrastructure applications, it's essential to diagnose and resolve issues promptly. By following the below steps, you can effectively troubleshoot and resolve issues with failed deployments in the infrastructure applications using Jenkins Pipeline and the Kubernetes Dashboard.

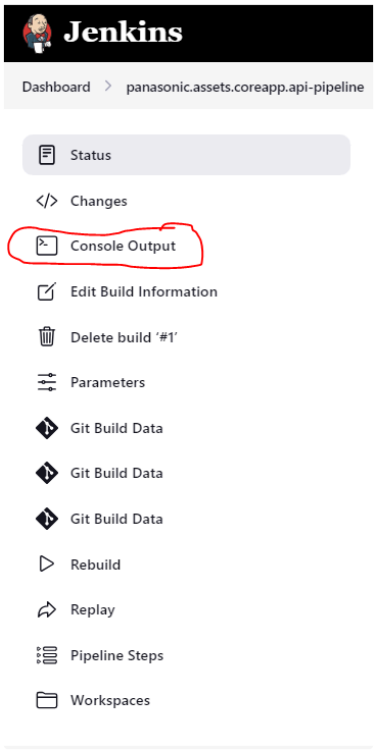
Check Jenkins Pipeline Logs

1. Open a web browser and go to http://<CB_HOSTNAME>:8080/
2. Log in with the username **admin** and password **admin**.

3. Navigate to the failed application's pipeline and select the failed build number.



4. Review the logs in the Console Output for detailed error messages.



Check Pods from Kubernetes Dashboard

1. Access the Kubernetes Dashboard by visiting https://<CB_HOSTNAME>:6443/api/v1/namespaces/kubernetes dashboard/services/https:kubernetes-dashboard:/proxy/#/login
2. Obtain a token for authentication.
3. Log in to the dashboard using the token.
4. Locate the failed Pod in the dashboard and review its logs for additional information.

Secrets

Secrets are sensitive pieces of information that need protection from unauthorized access. They are essential to the security of applications, adapters, and systems, granting access to various resources, systems, and data. Secrets typically include the following:

- **API keys**: Authentication tokens used to access APIs.
- **Passwords**: Credentials for user accounts, databases, or other services.
- **Certificates and private keys**: Used for secure communication between services.
- **Access tokens**: Temporary tokens used for session-based authentication.
- **Encryption keys**: Used to encrypt and decrypt data.

The secret templates are stored inside the repository listed below:

- https://bitbucket.org/panasonicdsc/panasonic.smartmanufacturing.configurations/src/master/secrets-template/

The secret files are stored in the Azure locations listed below:

- For Prod: https://portal.azure.com/#@pcus1.onmicrosoft.com/resource/subscriptions/37320104-2470-4423-8a65-4a6523845b67/resourceGroups/amcmainrg_prod/providers/Microsoft.Storage/storageAccounts/amcsecretssa0prod/storagebrowser
- For QA: https://portal.azure.com/#@pcus1.onmicrosoft.com/resource/subscriptions/ccd69e81-ed0f-410b-bc43-506f11a722b5/resourceGroups/smpksmainrg_qa/providers/Microsoft.Storage/storageAccounts/smpksmainsa0qa/overview

Checking the Secrets

Follow these steps to ensure the security of secrets:

1. Identify all places where secrets are stored or used, such as environment variables, configuration files, databases, source code, and secret management tools.
2. Use static analysis tools to scan your codebase for hardcoded secrets. Tools like GitGuardian, TruffleHog, and AWS's `git-secrets` can help identify secrets left in code inadvertently.
3. Review access logs from secret management systems (e.g., Azure Key Vault, AWS Secrets Manager) to check who has accessed or attempted to access secrets.
4. Ensure that secrets are stored in secure, centralized secret management systems like Azure Key Vault, AWS Secrets Manager, or HashiCorp Vault. These systems often provide auditing, access control, and automated rotation.
5. Check environment variables in your deployment environments to ensure secrets are correctly set and not accidentally exposed.
6. Review configuration files to ensure that secrets are not stored in plaintext. Use secret references or encrypted values instead.
7. Verify that secrets stored at rest (e.g., in databases, files) are encrypted using strong encryption standards.
8. Deploy automated secret detection tools that monitor your repositories and infrastructure for exposed secrets continuously. These tools can alert you when a secret is found in an insecure location.

Note: Always use secure and centralized tools to store and manage secrets.

Troubleshooting Problems with Secrets Management

Secrets can cause issues if they are compromised, misconfigured, or mishandled.

Follow these steps to detect and respond to problems related to secrets:

1. **Unauthorized Access**:
   - **Signs**: Unexpected access patterns in logs, unusual activity in systems, or breaches.
   - **Detection**: Monitor logs from secret management tools, applications, and network traffic. Set up alerts for abnormal access patterns or failed access attempts.
   - **Response**: Rotate the compromised secret immediately, revoke access tokens, and investigate the breach.
2. **Failed Authentication**:
   - **Signs**: Applications or services fail to authenticate or connect to external services, resulting in downtime or errors.
   - **Detection**: Monitor application logs for repeated authentication errors or connection failures. Use health checks and automated testing to catch these issues early.
   - **Response**: Verify that the secret (e.g., API key, password) is correct and hasn't expired or been rotated without updating dependent systems.
3. **Secret Exposure**:
   - **Signs**: Secrets accidentally committed to version control, exposed in logs, or included in error messages.
   - **Detection**: Use tools that scan for exposed secrets in code repositories and logs. Implement access control and review processes to catch accidental exposure.
   - **Response**: Rotate the exposed secret immediately, remove the exposed instance, and ensure it is securely stored going forward.
4. **Outdated or Expired Secrets**:
   - **Signs**: Services start failing due to expired certificates, tokens, or other time-sensitive secrets.
   - **Detection**: Monitor expiration dates and set reminders for secret rotation. Implement automation to refresh secrets before they expire.
   - **Response**: Update or renew the expired secret, and ensure dependent services are reconfigured with the new secret.
5. **Incorrectly Configured Secrets**:
   - **Signs**: Services behave unexpectedly due to incorrect secret values (e.g., wrong API key, wrong encryption key).
   - **Detection**: Perform regular configuration audits, automated tests, and integration tests to ensure that secrets are correctly configured.
   - **Response**: Correct the configuration, redeploy with the correct secret, and update documentation to prevent future issues.

Database Replication Issues

This section provides detailed troubleshooting instructions for resolving database replication issues between different environments (e.g., VIP-SH and QA). It focuses on a scenario involving an ALTER TABLE command on a large table (e.g., Transactions), which led to replication delays and application startup issues.

Symptoms

- Replication processes not functioning properly.
- Application startup migration unable to complete due to locks on the database.
- Large table sizes cause tools and scripts to hang.
- Significant size difference between environments, e.g., QA environment database size much smaller than production (VIP-SH).
- Alter table operation requiring exclusive locks, which blocks replication and other operations.

Root Causes

- **Table Size**: The Transactions table in the production environment is approximately 197.47 GB, while the same table in the QA environment is around 10.78 GB.
- **Alter Command**: The command `ALTER TABLE "Transactions" ALTER COLUMN "Quantity" TYPE numeric(28,10)` requires an exclusive lock, halting replication and other operations.
- **Replication Parameters**: Default PostgreSQL replication settings were insufficient to handle large datasets.
- **Application Constraints**: Readiness/liveness probes and default EF (Entity Framework) migration timeouts prevented completion of large migrations during application startup.

Troubleshooting Database Replication Issues

Follow these steps to debug and resolve the database replication issues:

1. Verify if the database version in the production environment (VIP-SH) matches the version in QA.
2. Identify the size of the affected tables in both environments.
   For example:
   - VIP-SH "Transactions" table: 197.47GB approx.
   - QA "Transactions" table: 10.78GB approx.
3. Confirm if the migration occurs during application startup.
4. Identify whether the migration script alters critical tables (e.g., "Transactions") and causes locks.
5. Modify the PostgreSQL configuration to optimize replication and large table handling:
   - Updated Parameters:
     `listen_addresses = '*'`
     `max_wal_senders = 10`
     `max_connections = 400`
     `shared_buffers = 10048MB`
     `cpu_tuple_cost = 0.03`
     `effective_cache_size = 16GB`
     `wal_keep_size = 4096`
     `max_wal_size = 10GB`
     `min_wal_size = 1GB`
     `wal_sender_timeout = 120s`

`wal_receiver_timeout = 120s`

`checkpoint_timeout = 10min`

`maintenance_work_mem = 1GB`

- Original Parameters for reference:
  `listen_addresses = '*'`
  `max_wal_senders = 10`

  `max_connections = 400`

  `shared_buffers = 2048MB`

  `cpu_tuple_cost = 0.03`

  `effective_cache_size = 7GB`

  `wal_keep_size = 4096`

6. Apply changes and restart the PostgreSQL service.
7. Purge old transaction data in the "Transactions" table to reduce size.
8. Consider using a data cleanup tool, but ensure it can handle large datasets without hanging.
9. Validate the impact on testing and ensure no critical data is lost.
10. Create a new column with the desired type (e.g., `numeric(28,10)`), copy data from the existing column to the new column, and drop the old column in a separate migration to avoid table locks.
11. Increase `initialDelaySeconds` for readiness/liveness probes temporarily during large migrations.
12. Disable Kubernetes probes during the migration to prevent pod restarts.
13. Update EF timeout in the migration context.
14. Verify replication is functioning as expected:
    - Check logs on DB pods.
    - Run tests to ensure synchronization between primary and replica databases.
15. Coordinate with QA and Performance team to run the verification scripts.
16. Log all changes to PostgreSQL parameters, application configurations, and migration strategies in a central repository (e.g., Jira ticket).

Fix

- **Optimize Table Storage**:
  - Avoid keeping dynamic data such as transactions in PostgreSQL for production environments.
  - Use external storage solutions (e.g., Druid) for dynamic data.
- **Plan for Large Migrations**:
  - Always validate table sizes and the number of records before deploying changes.
  - Schedule separate migration steps for large tables.
- **Proactive Cleanup**:
  - Regularly purge dynamic data from large tables to avoid blocking deployments.
  - Maintain scripts or tools for automated data cleanup.
- **Testing Environment Alignment**:
  - Ensure QA environments have similar data volumes to production for realistic testing.

Conclusion

After making the adjustments mentioned above, database replication can be successfully resumed, and the environment has been restored to a functional state. Documenting this issue serves as a valuable reference for future incidents related

to large-scale migrations and replication bottlenecks.

Troubleshoot Failed Deployment

If deployment fails in platform applications, it is crucial to diagnose and resolve issues promptly.
Follow these steps to troubleshoot failed deployments using Jenkins Pipeline and the Kubernetes Dashboard:

Troubleshooting Failed Deployments in the Jenkins Pipeline

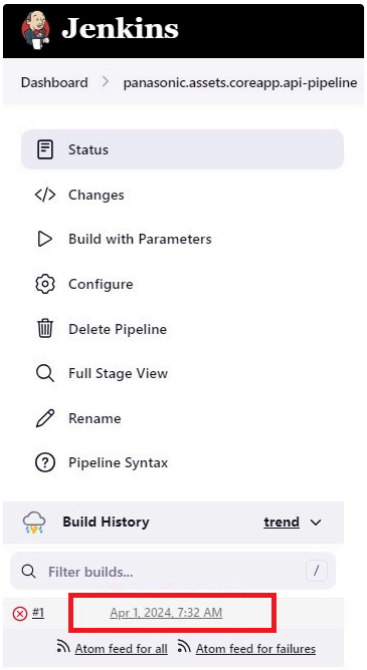
Follow these steps to troubleshoot failed deployments in the Jenkins pipeline:

1. Open a web browser and enter the Jenkins URL: http://<CB_HOSTNAME>:8080/.
   The Jenkins login page appears.
2. Enter username and password in their respective fields (admin/admin).

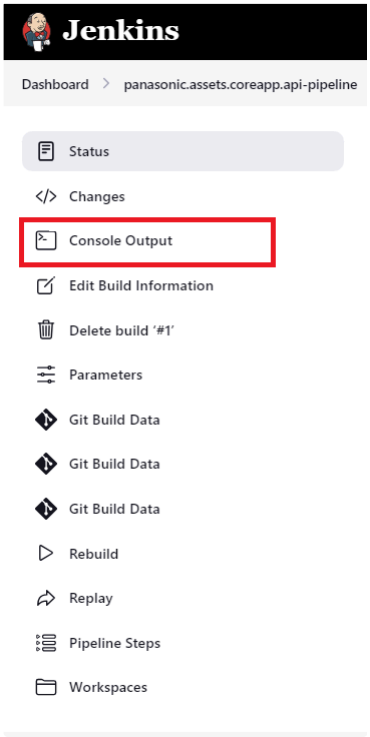
4. Click **Sign in**.
   The Jenkins homepage appears.
5. From the Jenkins dashboard, locate the pipeline or job to check logs.
   This can be found in the list of jobs on the main page or within a specific folder if the Jenkins instance uses folders.
6. Click on the name of the failed pipeline or job to check logs.

6. Click on the **build number** or **timestamp** of the specific build from the Build History.
   The build details page appears for the selected build number.

Note: The Build History section lists recent builds with their respective build numbers and status indicators (e.g., success, failure).



7. Click **Console Output** to view the detailed logs.
   The console output displays the log of the pipeline execution, showing all the steps executed, including any errors encountered.

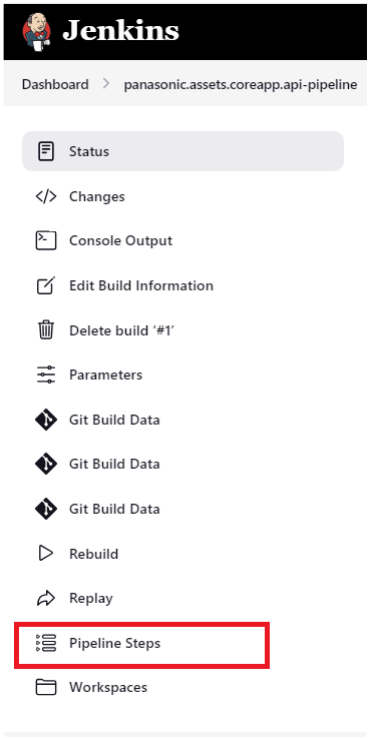


8. Check the log entries to identify error messages.
   These messages typically contain keywords like "ERROR", "FAILURE", "Exception", or specific error codes.
9. Analyze error messages to understand the nature of the error.

> Look for stack traces, specific error codes, or any other details that can provide insights into the cause of the error.

10. Click **Pipeline Steps** to check the pipeline configuration.
    Verify that the configuration for each stage is correct. This includes checking the script or commands being executed,

environment variables, and any other configurations.



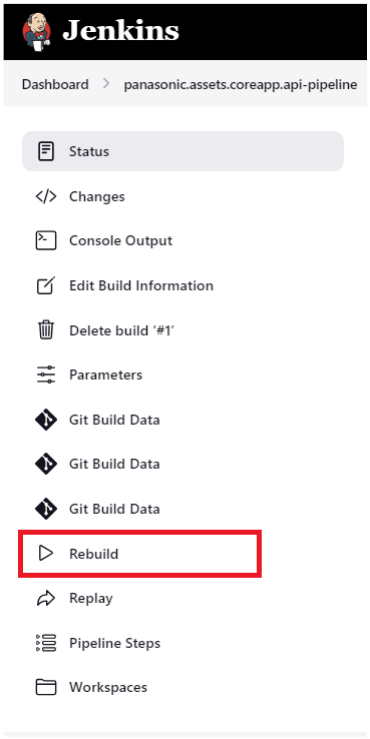
11. Verify the source code being built does not have errors.
    This may involve checking the repository for recent changes that might have introduced issues.

> **Note**: Verify that all dependencies are correctly specified and available.

12. Check the build environment (e.g., Jenkins agents, Docker containers) has all the necessary tools and dependencies installed.

> Check for any resource limitations (e.g., memory, disk space) that might be causing the build to fail.

13. If changes are needed, update as necessary, then click **Rebuild** to start a new build.

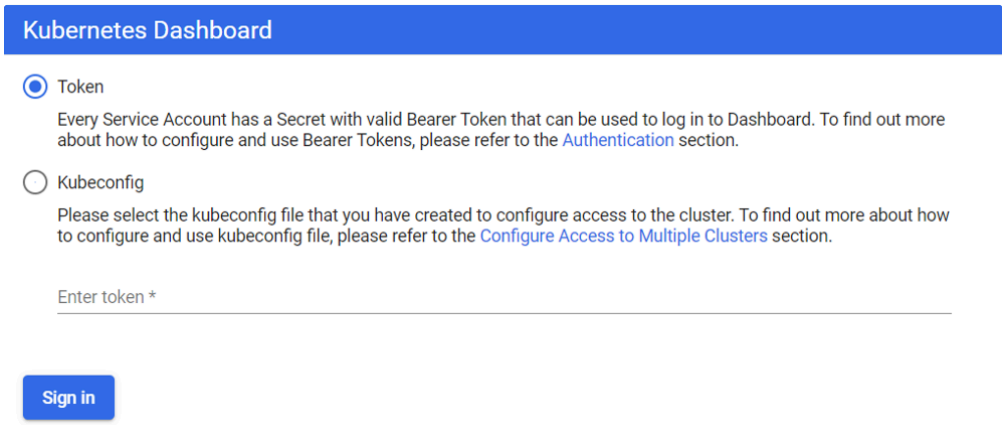


15. Click **Console Output** to monitor that the errors are resolved, and the build completes successfully.

Troubleshooting Errors in the Kubernetes Logs

Follow these steps to identify and rectify errors observed in the Kubernetes logs:

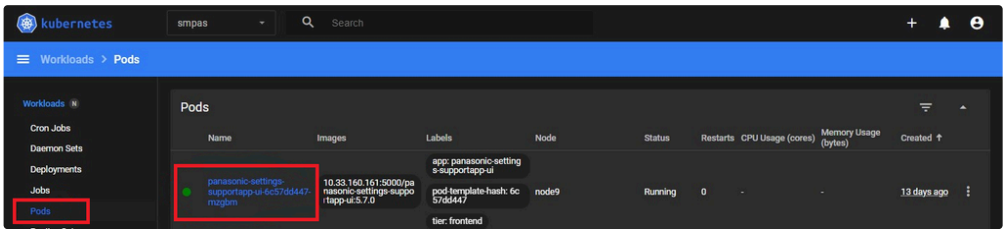
1. Open a web browser and enter the Kubernetes URL: https://<CB_HOSTNAME>:6443/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/#/login
   The Kubernetes Dashboard login page appears.
2. Select one of the options to enter the necessary credentials or authentication method (e.g., Token or Kubeconfig).



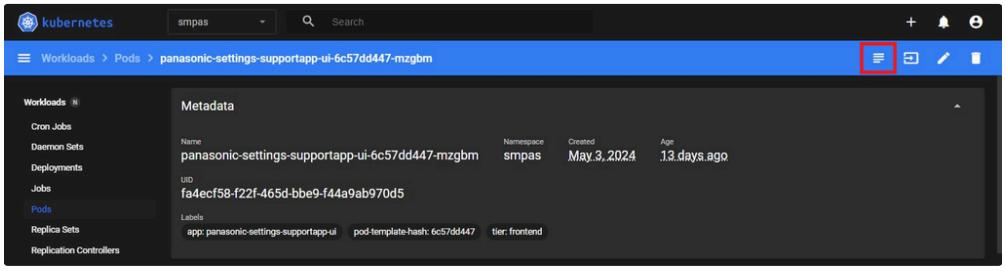
3. Click **Sign in**.
   The Kubernetes homepage appears.
4. Select the namespace from the dropdown list.

Note: If you are working with multiple namespaces, ensure that the correct namespace is selected from the namespace dropdown list.

3. From the navigation menu, Go to **Workloads** > **Pods**.
   The Pods page displays.
4. Browse through the list of pods or use the search bar at the top to locate the failed pod.
5. Click on the pod name to open its details.

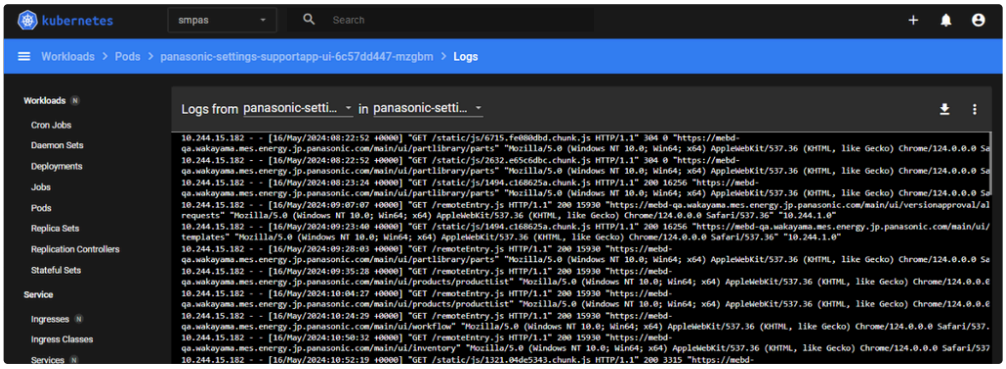


6. Click on the **Logs** icon at the top right of the pod details page.
   The logs display in a scrollable window. Each log entry will typically show the timestamp, the log level (if available), and the log message.



> **Note**: The logs icon typically looks like a document with lines representing logs.

8. Check the log entries to identify error messages.
   Errors typically have status codes like `500` (Internal Server Error), `404` (Not Found), or other HTTP status codes indicating issues. For example, the log entries with `HTTP/1.1" 500` indicate server-side errors.



> **Note**: Logs can be downloaded for offline analysis by clicking the download icon. The three-dot icon or options menu typically provides additional actions to perform on the current logs view.

9. Analyze error details by checking the URL endpoints, timestamps, and error codes.
   For example, entries like `[16/May/2024:08:10:24 +0000] "GET /remoteEntry.js HTTP/1.1" 500` indicate a failed request to `/remoteEntry.js`.
10. Check the application logs for more detailed error messages if available.
    This can provide insights into what is causing the server to return a `500` error.
11. Review the code corresponding to the failing endpoints. Check for issues such as unhandled exceptions, incorrect configurations, or resource constraints.

> **Note**: Ensure that all required services and dependencies are up and running.

12. Check the deployment configurations for `panasonic-settings-supportapp-ui` are correct.

> **Note**: Verify that the pod is correctly configured and has the necessary environment variables and secrets. Ensure that the related services and ingress configurations are correctly routing traffic to the pod.

13. Monitor the resource usage (CPU, memory) of the pod.

> **Note**: Resource exhaustion can cause errors and degraded performance. Adjust resource requests and limits in the deployment configuration if necessary.

14. If code changes are required, make the necessary updates and redeploy the application.

> Use CI/CD pipelines to automate the build, test, and deployment processes.

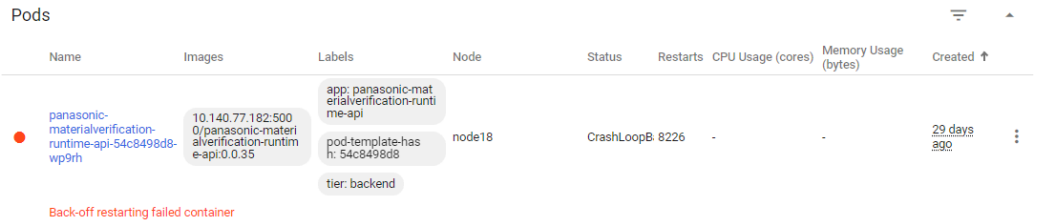
15. Test the application to ensure the errors are resolved.
16. Monitor the logs to confirm that the errors no longer appear, and the application is functioning correctly.

Crash Loop Backoff Error

The CrashLoopBackOff error in Kubernetes indicates that a container within a pod is repeatedly crashing and the system is backing off from restarting it too quickly. This is a common error that occurs when a container fails to start successfully, and Kubernetes makes multiple attempts to restart it, but the container continues to fail.

Follow these steps to rectify the CrashLoopBackOff error:

1. View the logs of the failing container to identify the cause of the crashes.



2. Check the events associated with the pod to get more details about the errors.
3. Check the containerized application for bugs or issues that may cause it to crash, such as uncaught exceptions, segmentation faults, or configuration errors.
4. Verify if the pod has sufficient CPU and memory resources allocated.
5. Ensure that the pod's configuration (such as environment variables, secrets, and config maps) is correct.
6. Confirm that the image `10.140.77.182:5000/panasonic-materialverification-runtime-api:0.0.35` is not corrupted and is working as expected.
7. Verify that all dependencies and services required by the application are available and functioning properly.

Logs

Troubleshooting Logs

Troubleshooting logs involves analyzing and resolving issues within the log files generated by a system or application. Logs are records of events that occur within a system, including errors, warnings, and informational messages. By examining these logs, you can identify problems, track down their root causes, and take corrective action.

Following are the steps to troubleshoot the logs:

Accessing Logs

- To access logs, open the Kibana dashboard using the provided base URL.
- Once logged in, navigate to the 'Logs' section or 'Discover' tab in Kibana to view the logs.

Filtering Logs

- Apply filters to narrow down the logs based on specific criteria such as time range, severity, or keywords related to the issue.
- Use the search bar to enter keywords or phrases that are relevant to the problem you are troubleshooting.

Analyzing Logs

- Look for error messages or warnings in the logs that indicate potential issues with the SmartHive platform.
- Pay attention to timestamps and sequences of events in the logs to understand the context of the issue.

Resolving Issues

- Once the issue is identified in the logs, take necessary actions to resolve it. This may involve restarting services, updating configurations, or fixing bugs in the code.

Monitoring Logs

- Keep monitoring the logs regularly to ensure that the issue has been resolved and no new issues have emerged.
- Set up alerts in Kibana to notify you of any critical errors or warnings in the logs.

Dashboards

Troubleshooting Dashboards

Troubleshooting dashboards involves identifying and resolving issues related to the visualization and functionality of dashboards in the SmartHive platform. Following are the detailed steps on how to troubleshoot dashboards effectively:

Accessing the Dashboard

- Access the Kibana dashboard using the URL_____
- Ensure that you have the necessary permissions to view and interact with the dashboard.

Dashboard Loading Issues

- If the dashboard is not loading or is taking too long to load, check the network connectivity to ensure that you can reach the Kibana server.
- Verify that the Kibana server is up and running without any issues.

Monitoring Service Health

- Use the Kibana dashboard to monitor the health of the services that are being visualized.
- Look for any anomalies or errors in the data being displayed on the dashboard.

Identifying Dashboard Issues

- If the dashboard is behaving unexpectedly or displaying incorrect data, check the data sources to ensure that they are providing accurate information.
- Verify that the queries and filters applied to the dashboard are correct and are not causing the issues.

Refreshing the Dashboard

- Sometimes, simply refreshing the dashboard can resolve minor display issues.
- Use the refresh button on the dashboard interface to reload the data.

Checking Data Sources

- If the dashboard relies on external data sources, check the status of these sources to ensure that they are accessible and providing the expected data.
- Look for any errors or timeouts in the data retrieval process.

Clearing Cache

- Clear the browser cache and cookies to ensure that you are viewing the latest version of the dashboard.
- Sometimes, cached data can cause display issues.

Rebuilding the Dashboard

- If the dashboard continues to have issues, consider rebuilding it from scratch.
- This can help resolve any underlying issues with the dashboard configuration.

Seeking Support

- If you are unable to resolve the dashboard issues on your own, seek support from the dashboard administrators or technical support team.
- Provide detailed information about the issue, including any error messages or unusual behavior observed.

Solution Applications

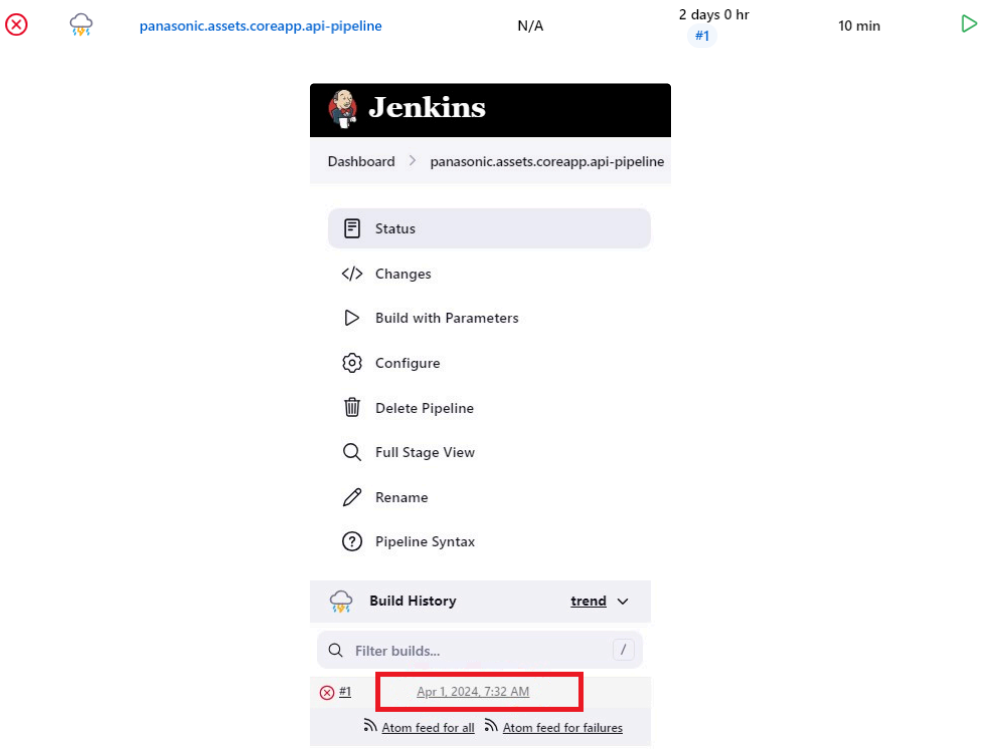
Troubleshooting Steps for Failed Deployments in Solution Applications

When deployments fail in Solution Applications, it's essential to diagnose and resolve issues promptly. By following the below steps, you can effectively troubleshoot and resolve issues with failed deployments in the Solution Applications using Jenkins Pipeline and the Kubernetes Dashboard.

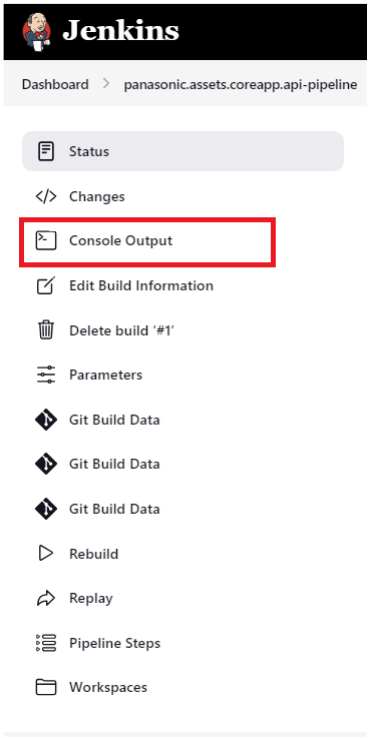
Check Jenkins Pipeline Logs

1. Open a web browser and go to http://<CB_HOSTNAME>:8080/
2. Log in with the username **admin** and password **admin**.

3. Navigate to the failed application's pipeline and select the failed build number.



4. Review the logs in the Console Output for detailed error messages.



Check Pods from Kubernetes Dashboard

1. Access the Kubernetes Dashboard by visiting https://<CB_HOSTNAME>:6443/api/v1/namespaces/kubernetes dashboard/services/https:kubernetes-dashboard:/proxy/#/login
2. Obtain a token for authentication.
3. Log in to the dashboard using the token.
4. Locate the failed Pod in the dashboard and review its logs for additional information.

Troubleshooting Workflows

Troubleshooting workflows is crucial for ensuring the smooth operation of processes in the SmartHive platform. Following are the steps required to troubleshoot workflows effectively:

Accessing Workflow Runtime

- Navigate to the workflow runtime section of the SmartHive platform.
- Ensure that you have the necessary permissions to view and manage workflows.

Monitoring Workflow Execution

- Use the workflow runtime to monitor the execution of workflows in real time.
- Check the status of each workflow and look for any workflows that are stuck or failing.

Viewing Workflow Logs

- Access the logs related to workflow execution from the runtime interface.
- Look for any error messages or warnings that indicate issues with the workflow.

Applying Filters

- Apply filters to the logs to narrow down the search and focus on specific workflows or types of issues.
- Filter logs based on workflow ID, status, timestamp, or any other relevant criteria.

Identifying Root Causes

- Analyze the logs to identify the root cause of any workflow issues.
- Look for patterns or recurring issues that may indicate underlying problems with the workflow design or execution environment.

Resolving Workflow Issues

- Once the root cause is identified, take appropriate action to resolve the workflow issues.
- This may involve modifying the workflow design, fixing configuration issues, or addressing any environmental issues.

Testing and Validation

- After making changes to resolve the workflow issues, test the workflow to ensure that the problem is resolved.
- Validate the workflow by running it in a test environment and verifying that it behaves as expected.

Determining Changes in Workflows

Follow these steps to determine what changed in workflows that caused them to transition from a working state to a non-working state (or vice versa):

1. **Identify the Workflow in Question**

- Clearly identify which workflow or part of the workflow has changed behavior.
- Determine the expected vs. actual behavior to understand what specifically is "not working."

2. **Review Version History and Logs**

- **Version Control**: Check the version history of the workflow if it's tracked in a version control system (e.g., Git). Look for recent changes to the workflow definition, scripts, or configuration files.

- **Logs**: Examine logs from the system where the workflow is running. Logs may provide timestamps and details about what changed, error messages, or warning signals around the time the workflow stopped working.

3. **Compare Configurations**

- **Environment Variables**: Compare the environment variables in the working state and the non-working state. Configuration drift (changes in environment variables) is a common cause of workflow issues.
- **Infrastructure Changes**: Look at changes in the infrastructure that the workflow depends on, such as changes in servers, databases, or network configurations.

4. **Check Dependencies and Integrations**

- **External Services**: Identify if any external services or APIs that the workflow depends on have been updated, deprecated, or have had their credentials changed.
- **Internal Services**: Check for changes in internal services or microservices that the workflow interacts with. This could include updates to databases, microservices, or data schemas.

5. **Audit Recent Changes**

- **Code Changes**: Review recent code changes that might have affected the workflow, especially any changes in related scripts, libraries, or modules.
- **System Updates**: Check if there were any recent updates to the operating system, middleware, or other platform components that could affect the workflow.
- **Security Updates**: Look for changes related to security patches or updates, as these can sometimes restrict operations that were previously allowed.

6. **Test with Known Good Configurations**

- **Rollback**: If possible, roll back to a known good configuration or version of the workflow. This can help isolate the change that caused the issue.
- **A/B Testing**: Run the workflow in a controlled environment with both the old and new configurations to identify where the behavior diverges.

7. **Analyze Data Inputs and Outputs**

- **Data Changes**: Examine the input data to the workflow. Changes in data structure, format, or content could cause the workflow to fail.
- **Output Comparison**: Compare the outputs from when the workflow was working to the outputs when it wasn’t. This might help you identify at what point things are going wrong.

8. **Use Automated Tools**

- **Monitoring and Alerts**: Use monitoring tools that track workflow performance and can alert you to specific changes in behavior.
- **CI/CD Pipelines**: Check Continuous Integration/Continuous Deployment (CI/CD) pipelines for recent deployments or changes that might have affected the workflow.

9. **Reproduce the Issue**

- **Replication**: Try to reproduce the issue in a test environment. By systematically enabling or disabling features or reverting changes, you can often pinpoint the cause.
- **Step-by-Step Execution**: Break down the workflow into smaller steps and execute them manually to identify where the issue starts.

10. **Check for External Factors**

- **Network Issues**: Ensure there are no external factors such as network issues, third-party service outages, or resource limits (like memory or CPU) that might be affecting the workflow.

- **Time-Based Triggers**: Verify if the issue is related to time-based triggers, like cron jobs or scheduled tasks, and whether those triggers have been altered or misconfigured.

11. **Implement and Monitor Fixes**

- **Apply Fixes**: Once you've identified the change that caused the issue, apply the necessary fix or rollback the change.
- **Monitor**: After implementing the fix, closely monitor the workflow to ensure that it returns to a stable state and that no new issues arise.

By systematically comparing the working state to the non-working state and analyzing all potential changes, you can effectively identify what changed in your workflows and resolve the issue.

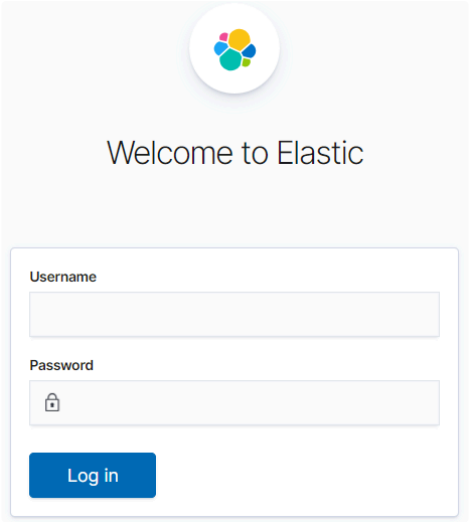
Checking Logs in Kibana

Use the following Kibana features to view and query logs ingested into Elasticsearch.

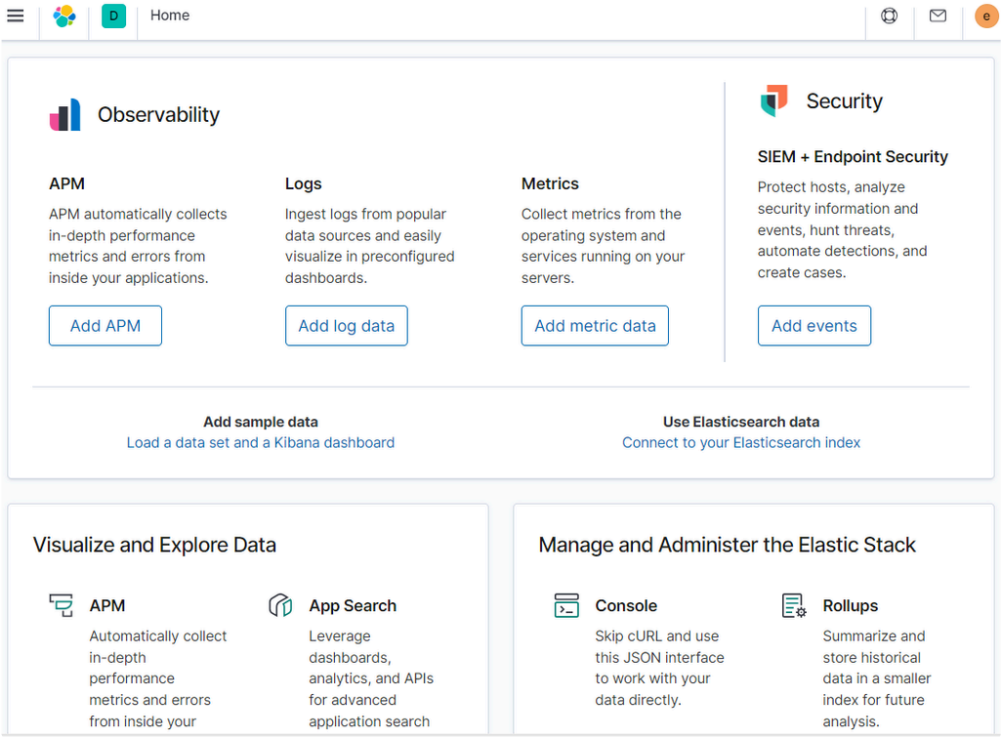
- **Logs UI** is used for exploring logs. It enables you to search, filter, and tail all the logs you have ingested into Elasticsearch.
- **Discover** focuses on rapid consumption of Elasticsearch data, including logs, with a standardized query language.

Follow these steps to check logs in Kibana:

1. Open a web browser and enter the URL: https://<customer domain name>/kibana/app/home#/ to access the Kibana (e.g. https://vip-sh.panasonicfa.com/kibana/app/home#/).
   The Elastic login page appears.
2. Enter your username and password in their respective fields.

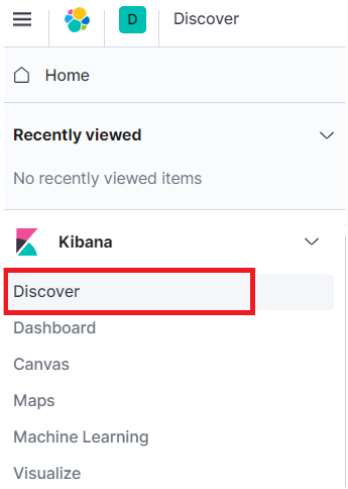


3. Click **Log in**.
   The Kibana homepage opens.

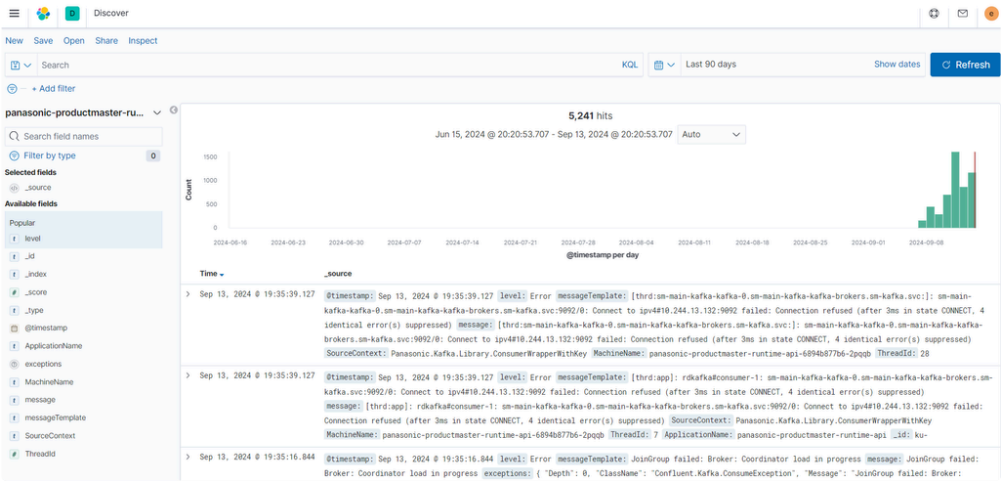


4. In the left-hand menu, click **Discover** from the dropdown list.

   The Discover page opens.



5. Select the appropriate index pattern from the dropdown. This determines which set of logs you're going to query.



6. At the top of the screen, set the time range to narrow down the logs to the relevant period. You can choose from predefined ranges or set a custom time frame.

Use the search bar to filter logs based on specific criteria. You can search by keywords, error codes, or other log attributes. The log entries matching your query will be displayed below the search bar.

7. Click on any log entry to expand it and view detailed information.
8. To refine your search further, you can add filters by clicking on any field in the log entry. This will add the filter to your search query.
9. Click **Save** at the top to save the query.

Note: If needed, you can export the logs to a CSV file for further analysis outside of Kibana. Click the “Share” button and select “CSV Reports.”

Troubleshooting IF Adapters

Logging

In order to effectively monitor and debug the adapter, it is essential to configure logging to use console logging. Standardized log implementation will not only be helpful in troubleshooting any issues with the adapter but will also prevent unnecessary information from being logged when the adapter is running fine.

For standardization, the adapter must implement the following three log levels:

- Debug
- Information
- Error

Information should be set as the default level. The table below lists the log levels along with guidelines on what to log in each level and the type of data that must be recorded. It is recommended to use each level appropriately according to this reference table. The table serves as a guideline and not as a mandatory guide on what to log. Other types of data and information should also be logged depending on the purpose of the adapter.

Log Level	Log Type	Log Data
Debug	Database Query Request / Response	The query and the response
	Message Broker Message Sent/ Received	The message broker message payload and response
	SMTP Request/Response	The SMTP message containing body and path and response
	Incoming/Outcoming Adapter Requests/Responses	The endpoint requested, status and body payload request and response
	Third party API Request/Response	The endpoint requested, status and body payload and the response
Information	Database connection established/closed	The path and name of database and status if applicable
	Message broker connection established/closed	The path and name of the broker
	SMTP server connection established/ closed	The path and name of the server and status if applicable
	Third party Socket connection established/ closed	The path and name of the server and status if applicable

Error	Handled/Unhandled adapter rules exceptions	Exception Message and Stack Trace information
	Database Timeout/ unreachable	Exception message, base path and stack trace information
	SMTP Timeout/ unreachable	Exception message, base path and stack trace information
	Third Part API Timeout/ unreachable	Exception message, base path and stack trace information
	Third Part Socket Timeout/ unreachable	Exception message, base path and stack trace information

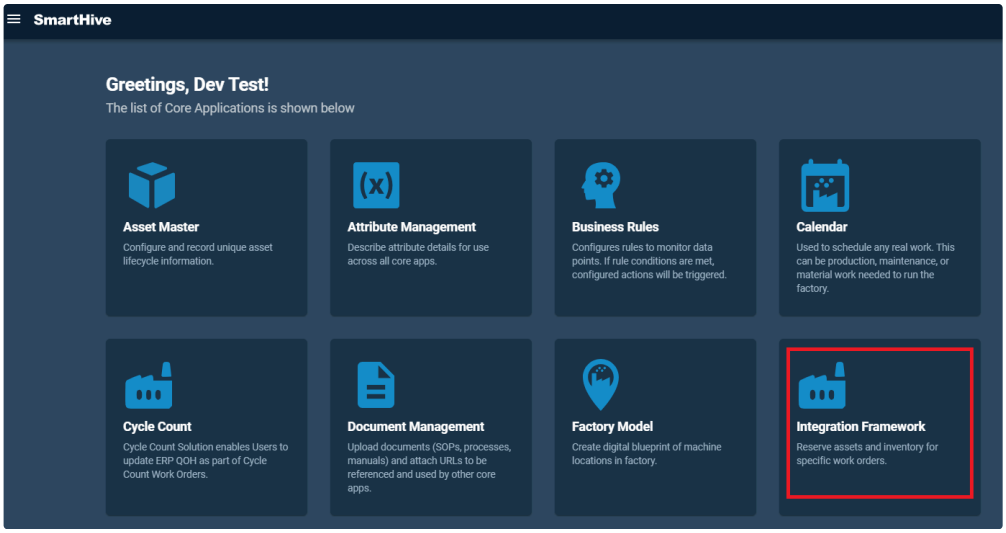
Checking Logs

Always start by checking the system and adapter logs for any error messages or warnings that can provide insights into the issue.

Follow these steps to check the logs:

1. Open a web browser and enter the URL: https://<customer domain name>/main/ui to access the SmartHive (e.g. https://sfdev.panasonicfa.com/main/ui).
   The SmartHive login page appears.
2. Enter your username or email ID and password in the respective fields.

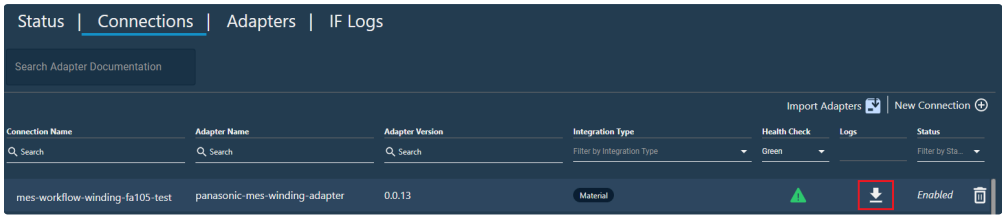
3. Click **LOG IN**.
   The SmartHive dashboard opens.
4. Click on the **Integration Framework** tile.
   The IF Manager interface opens.



5. Click on the **Connections** tab where you can manage adapters.

Note: If a connection is deleted, the logs are lost, and there will be no history of logs.

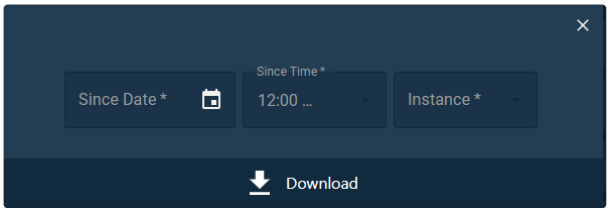
6. Click on the download icon to download logs of the adapter.



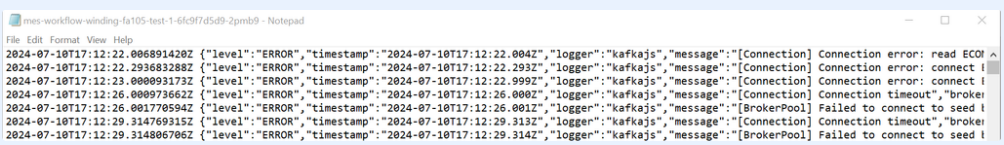
The "**Health Check**" feature allows users to quickly assess the health of various connections in the system and take necessary actions based on the status. The following options are available in the drop-down menu:

- **Grey:** Likely indicates that the health status of the connection is unknown or not yet assessed.
- **Green:** Indicates that the connection is healthy and functioning as expected.
- **Yellow:** Suggests that the connection might have some warnings or minor issues that need attention.
- **Red:** Indicates that the connection has significant issues or is failing, requiring immediate attention.

7. Select the following:
   - Date
   - Time
   - Instance
8. Click **Download**.



The details of the error will be found in a notepad file.



Resolving Log Errors

The most common errors are connection errors and timeouts, primarily related to KafkaJS, which is a Kafka client for Node.js. Follow these steps to resolve common log errors:

1. **Connection Error:** These errors indicate that the client is either having its connection reset (ECONNRESET) or being refused (ECONNREFUSED) when attempting to connect to the Kafka broker.

   `"Connection error: read ECONNRESET"`

   `"Connection error: connect ECONNREFUSED"`

   - **Potential Fixes:**
     - **Check Kafka Broker Availability:** Ensure that the Kafka broker is running and reachable from the client. Verify the broker's IP address, port, and network connectivity.
     - **Firewall/Security Groups:** Check if any firewalls or security groups are blocking the connection to the Kafka broker.
     - **Broker Configuration:** Ensure that the Kafka broker is correctly configured to accept connections on the specified port and that it's not overloaded.
2. **Connection Timeout:** The connection is timing out, which means the client is unable to establish a connection within the expected time.

   `"Connection timeout"`

   - **Potential Fixes:**
     - **Increase Timeout Settings:** Try increasing the connection timeout settings in your KafkaJS client configuration.
     - **Network Latency:** Investigate if there is network latency or instability causing the connection to time out. Optimize the network conditions if possible.
     - **Broker Performance:** Check the performance and resource usage of the Kafka broker. It might be under heavy load and unable to respond to connection requests in a timely manner.
3. **Broker Pool Errors:** This error indicates that the client is unable to connect to any brokers in the broker pool (often the initial brokers that the client contacts to get metadata).

   `"BrokerPool Failed to connect to seed"`

   - **Potential Fixes:**
     - **Verify Broker List:** Ensure that the broker list in the KafkaJS client configuration is correct and includes all the necessary seed brokers.
     - **Check Broker Status:** Verify that the seed brokers listed in the client configuration are up and running.
     - **Cluster Configuration:** Ensure that the Kafka cluster configuration is correct, and the brokers can communicate with each other.

Custom UI

The Custom UI feature allows users to design and configure personalized interfaces that cater to specific use cases or workflows. This flexibility lets users create tailored dashboards and interactive elements to visualize data, manage tasks, and improve productivity within the platform. This feature enhances the user experience by allowing a more hands-on and customized interaction with data and tools in SmartHive. The purpose of a custom UI is to create a common interface that can be utilized across multiple applications.

The key aspects of Custom UI include:

- **Custom Widgets**: Users can create and arrange widgets, such as charts, forms, and data tables, to display relevant data and actions.
- **Drag-and-Drop Design**: The interface supports drag-and-drop functionality, making it easy to design without needing coding skills.
- **Integration with Data**: Custom UIs can pull data from various sources within SmartHive, making it possible to interact with live data in real-time.
- **Automation & Triggers**: Users can set up actions and workflows to be triggered based on specific conditions or interactions within the custom UI.
- **User Roles & Permissions**: Custom UIs can be configured to show different views and options based on user roles or access permissions.

Creating a Custom UI

Follow these steps to create a custom MicroUI:

1. Create a component within the specified application (for example, react.js).
2. Expose the component to make it available to other applications.

> The links for the exposed components are stored in the remote.js folder for use.

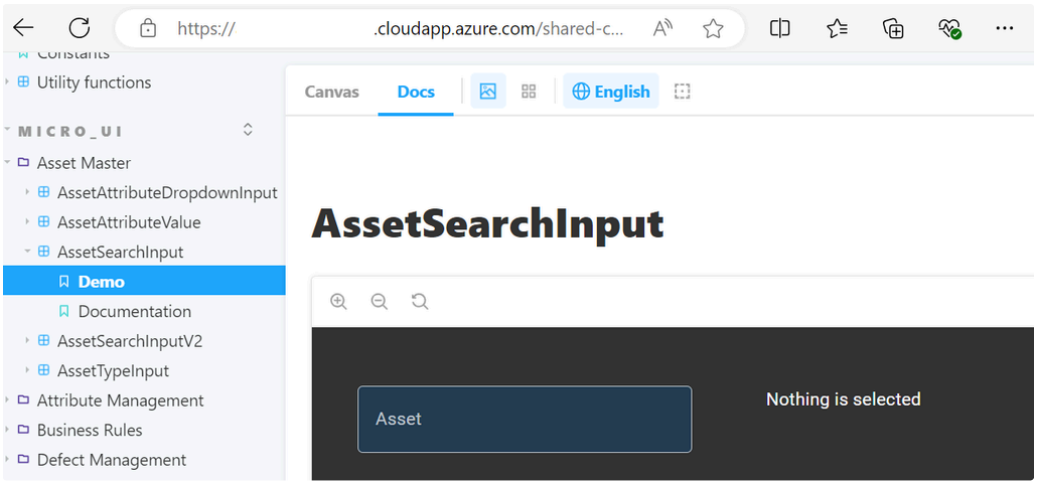
Below is a typical example of the **AssetSearchInput** component created within the **Asset Master** application:

> **Note**: To use MicroUI, you need to install version 2.3.0 of the library.
>
> `npm install @panasonic/shared-components-ui@latest`

1. To enable the MicroUI to access its API endpoints, you need to add these endpoints to your application's permissions configuration. Below is the list of endpoints used by the MicroUI:
   `[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object]`
2. Add the Asset Master URL variable to the .env file in your core application. This is the URL where the MicroUI will be loaded from. The code below points to the standalone version of the Asset Master, so you don't need to run Asset Master locally.
   `REACT_APP_SM_ASSET_MASTER_URL='https://amcapidev.eastus.cloudapp.azure.com/assetcoreapp/ui'`
   If, for any reason, you want to load the MicroUI from a locally running Asset Master, then the URL will look something like this:

   `REACT_APP_SM_ASSET_MASTER_URL='http://localhost:????'`

   Replace `????` with the port on which you're running Asset Master application.



3. Add corresponding lines to the module federation config in your core application:

```
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
[object Object]
```

   Most likely you will be able to find this config file in `scripts/overrides` folder.

4. Add Asset Master URLs proxy to the setupProxy.js file in your core application. If you're running your core application through the main container application, then you need to make sure that same lines are present in its `setupProxy.js` file as well.

```
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object]
```

> **Note**: You most likely already have a configured `setupProxy.js` file with `PROXY_URL` variable in your core application. But if not - check out how the main container application does it here. Remember to make sure that you have this variable in your `.env` file as well
> `REACT_APP_PROXY_URL='https://amcapidev.eastus.cloudapp.azure.com'`.

5. Add the MicroUI component declaration to make your TypeScript happy like this:

```
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object]
```

> If you don't know where you should add this code, then the easiest way of achieving this is to create a `[src/external.d.ts]` file and add the code there.

6. Using the MicroUI, you can import `AssetSearchInput` anywhere in your core application like this:

```
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],
```

`[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object]`

Note: MicroUI is not wrapped in `React.Suspense` internally. Consumer application should decide how to wrap the MicroUI and which fallback component to use.

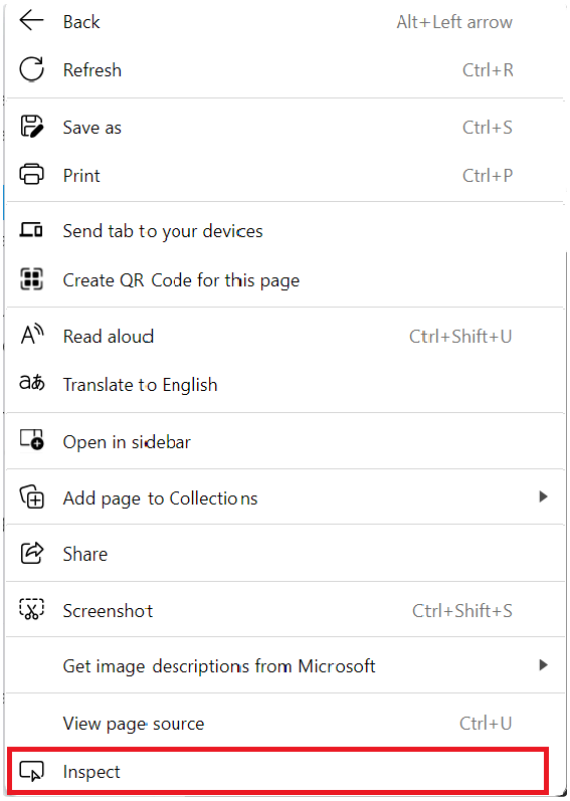
Typescript: MicroUI provides its interfaces and types via `AssetSearchInputTypes` namespace to avoid naming conflicts between consumer application and the MicroUI itself.

Troubleshooting Custom UI

Follow these steps to identify and report issues with the Custom UI:

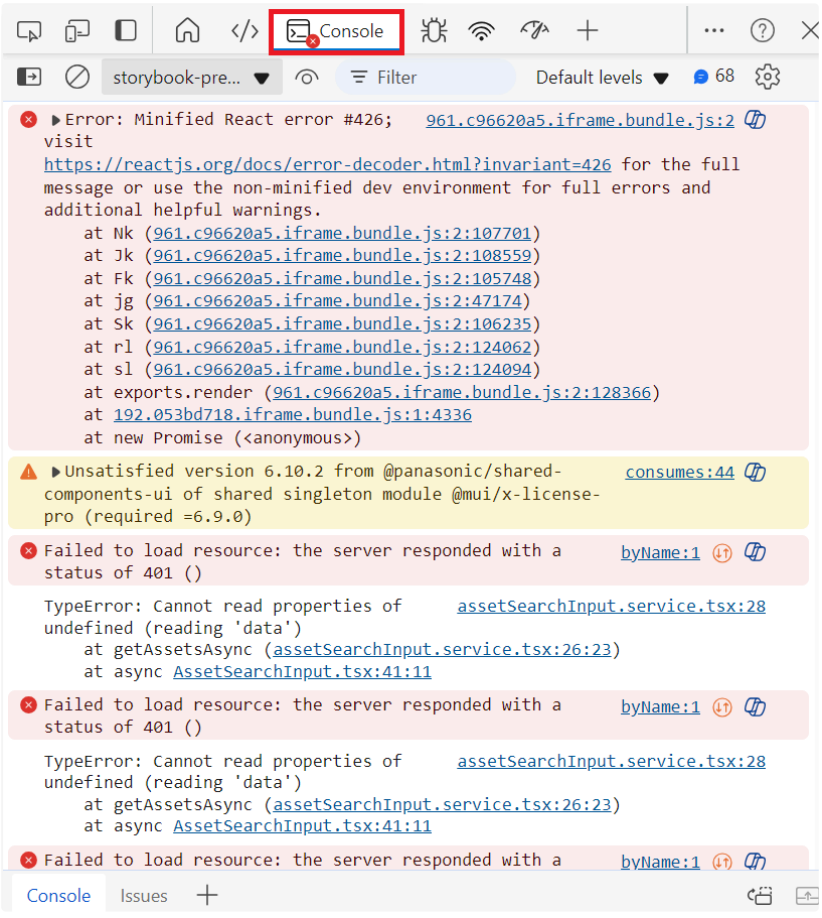
1. Open a web browser and enter the URL: https://<customer domain name>/main/ui to access the SmartHive (e.g. https://sfdev.panasonicfa.com/main/ui).
   The SmartHive login page appears.
2. Enter your username or email ID and password in the respective fields.

3. Click **LOG IN**.
   The SmartHive dashboard opens.
4. Click on the application tile you want to check.
   The user interface of the application opens.
5. Right-click anywhere on the UI page and select **Inspect**.

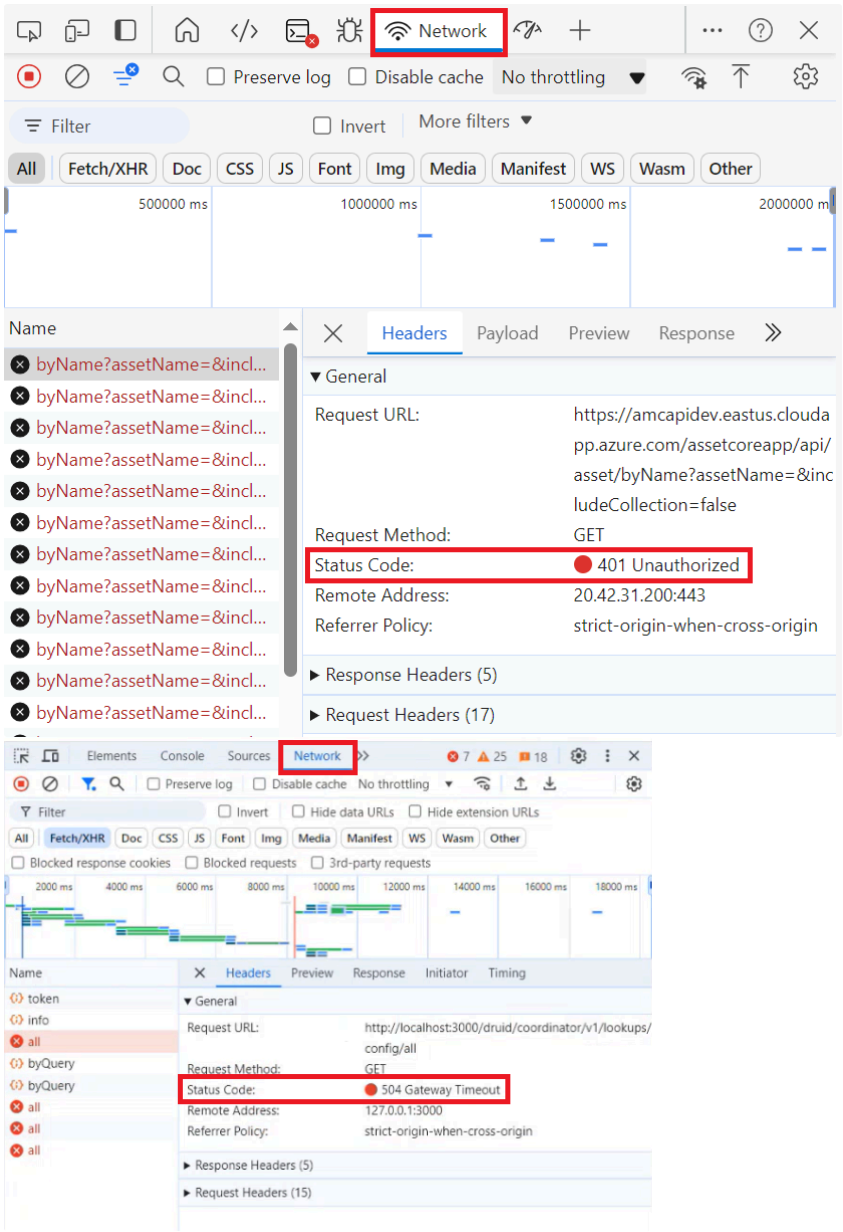


If there is a problem, the error will be displayed directly in the user interface.

6. Click on the **Console** tab to view error messages.



7. Click on the **Network** tab to view any issues related to the API.

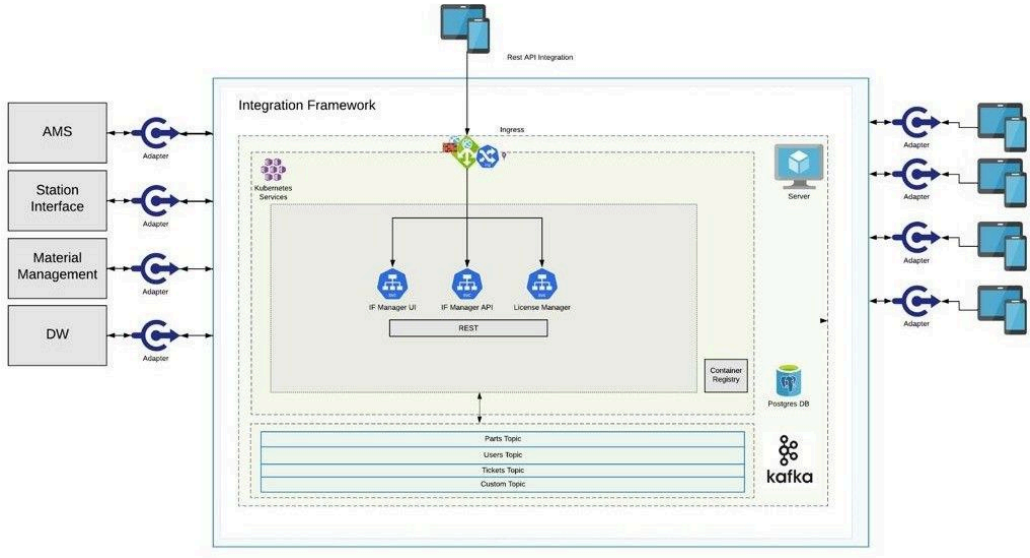


8. Review the error message, analyze it, and contact the developers for assistance.

> **Note**: The UI failures should be reported to the UI team (front-end team), while API failures should be reported to the API team (back-end team).

Adapters for IF Manager

Adapters for the Integration Framework Manager (IF Manager) are crucial components that allow the IF ecosystem to communicate with various applications and systems. They are standalone pieces of code, configured through environment variables, that solve specific integration challenges. Adapters are wrapped in Docker images, making them portable and easy to manage.



Adapters are a critical part of the IF ecosystem, allowing users to integrate various applications and systems seamlessly. By following best practices for adapter development and deployment, developers can ensure their adapters are efficient, reliable, and easy to manage.

Following are the list of key concepts:

Adapter Parameters

Parameters are configurations that make adapters flexible and maintainable. They are displayed in the IF Manager UI and can include values such as database host addresses, URLs, or Kafka configurations. Parameters are defined in a JSON manifest file and are configurable by the IF Manager user.

Connection

A connection is a running instance of an adapter with a specific configuration. Multiple connections can be created from a single adapter, each with its own set of parameters. Connections can be managed in the IF Manager's 'Connections' tab.

Adapter Versioning

Adapters are versioned to allow flexibility in the development process and to prevent breaking changes in existing connections. Users can select the adapter version when creating a connection, and different versions can run simultaneously.

Developing Adapters

Adapters can be developed in any programming language that can be dockerized and run in a Linux environment. Developers must follow standards to allow the adapter to be managed by IF, including defining parameters, configuring them in the JSON manifest, and recovering them through environment variables in the adapter code.

Integration with IF

Adapters must be able to read environment variables provided by IF, such as HOSTNAME, CONNECTION_ID, ADAPTER_TYPE_ID, and ENVIRONMENT_SETTINGS. These variables contain settings necessary for the adapter to communicate with IF, such as the events URL. Adapters can send messages to the events endpoint, which are displayed in the IF Manager UI's Status tab.

Deployment

To deploy a new adapter or a new version of an adapter, developers must follow specific steps:

1. Build a Docker image for the adapter.
2. Save the image as a .tar.gz file.
3. Create a JSON manifest file with metadata about the adapter or version.
4. Compress the .tar.gz and JSON files into a .zip file.
5. Import the .zip file into IF Manager using the Import Adapters feature.

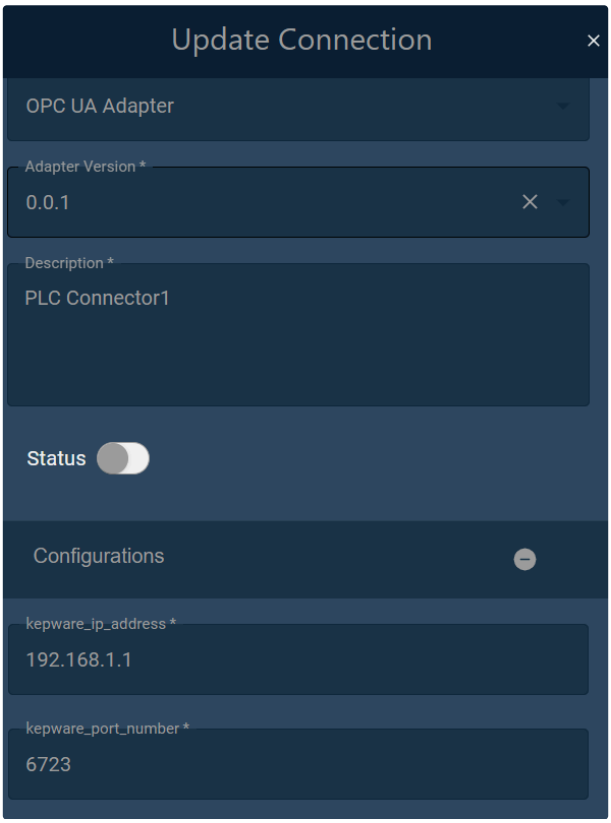
Configuring Adapters in IF Manager

The Integration Framework (IF) is a technology-agnostic framework designed to provide a standard for interaction and communication between adapters developed by third parties or Panasonic development teams. IF contains no business logic and serves as a middleware layer between applications. The Integration Framework Manager (IF Manager) is a user interface that allows field engineers or operations teams to manage IF instances, import adapters and new adapter versions, manage connections, check status, and download logs.

Following are the set of aspects used in Configuring Adapters in IF Manager:

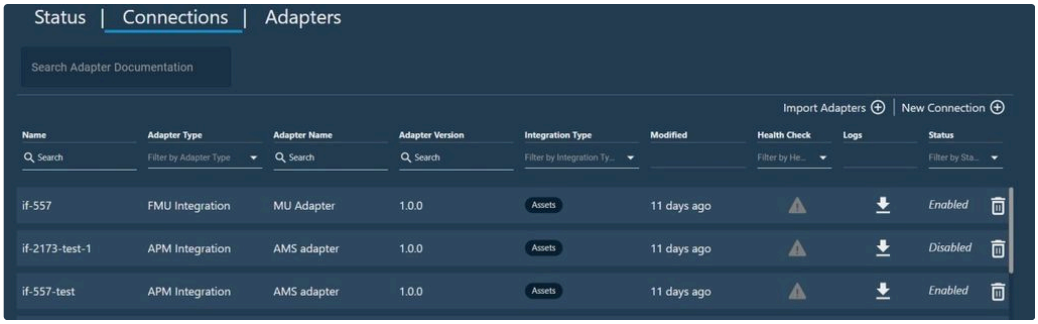
Adapter: A stand-alone piece of code written in any language, configured through environment variables, and wrapped in a Docker image. It is a code artifact that solves a specific problem.

Parameters: Configurations that make adapters flexible and maintainable. They are displayed in the IF Manager UI in the 'Configurations' accordion.



Parameters are displayed in the UI following the order they're defined in the JSON manifest of the adapter.

Connection: A running instance of an adapter with a specific configuration provided by the IF Manager user.



Adapter Versioning: New changes and updates in the adapter code require a new version of the adapter to be generated to avoid breaking changes in existing connections.

Developing Adapters

Adapters are Docker images that can run in the Linux operating system, allowing development teams to choose any programming language. Adapters must adhere to certain standards to be managed by IF, including defining parameters, configuring them in the JSON manifest, and recovering them through environment variables in the adapter code.

Environment Variables

Every adapter must be able to read environment variables like HOSTNAME, CONNECTION_ID, ADAPTER_TYPE_ID, and ENVIRONMENT_SETTINGS. These variables provide settings to communicate with IF.

Integrating with IF

Adapters can send messages to IF through an events endpoint, which serves as a channel between connections and IF. The endpoint allows connections to send messages that will be displayed in the Status tab of the IF Manager UI.

Deployment

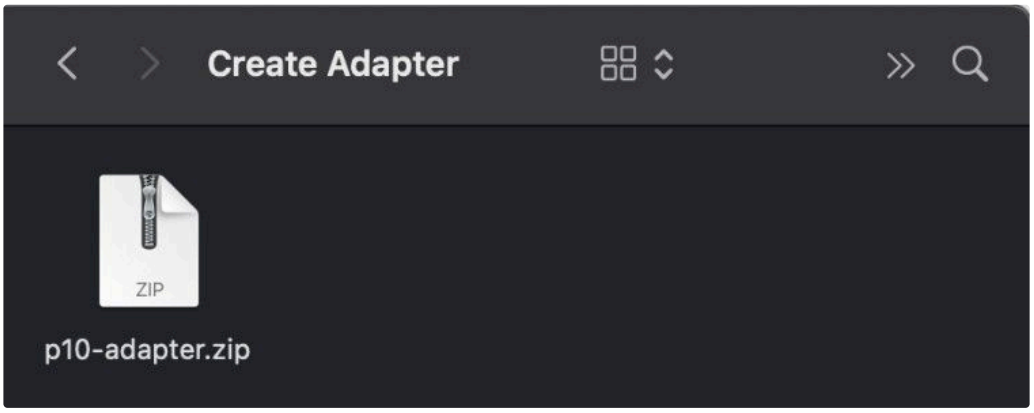
To deploy a new adapter or a new adapter version, developers must build a Docker image, save it as a .tar.gz file, and create a JSON manifest file with metadata about the adapter. The JSON file must follow specific rules regarding keys, values, and formats. After preparing the Docker image and JSON file, developers must compress them into a .zip file and import the adapter using the IF Manager UI.

Connect to Third Party software?

Integration Framework (IF) provides a standard for connecting to third-party software through adapters. Adapters are configured with environment variables and run as Docker containers in a Linux environment. This guide explains how to connect to third-party software using IF Manager.

Configuring Adapters for Third-Party Software

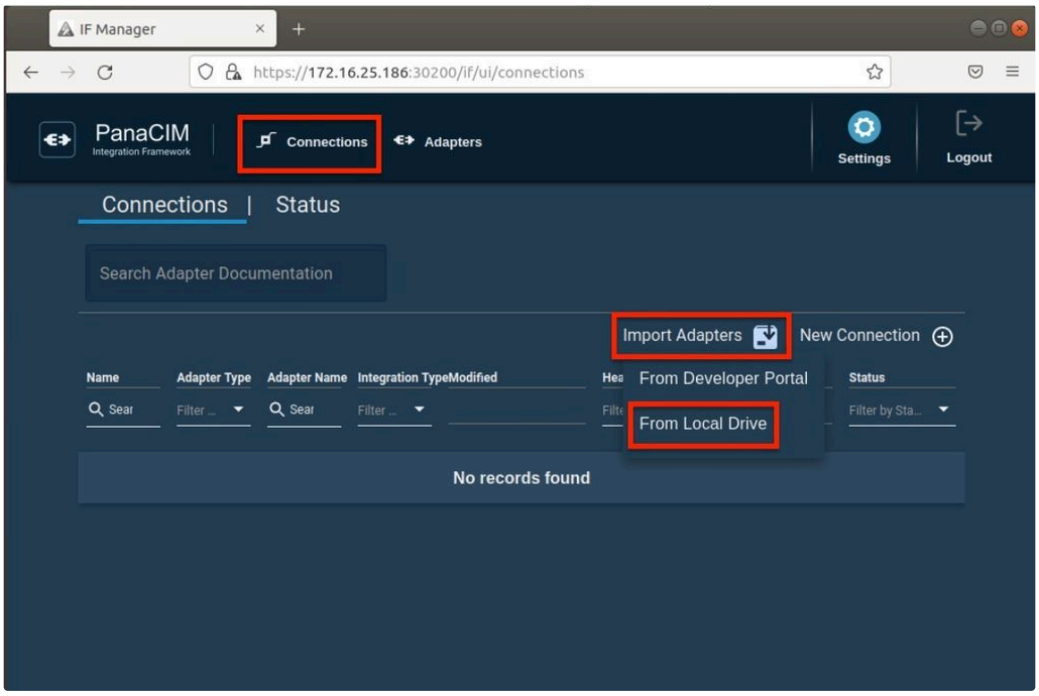
- **Build Your Docker Image:** Create a Dockerfile and build an image for your adapter. Ensure the image exposes port 80.
- **Save as .tar.gz:** Save the Docker image as a .tar.gz file.
- **Create a JSON File:** Create a JSON manifest file with metadata about the adapter, including its name, version, display name, adapter type, and attributes.
- **Zip tar.gz and JSON:** Compress both the .tar.gz and JSON files into a .zip file. Ensure they are at the root level of the zip file.



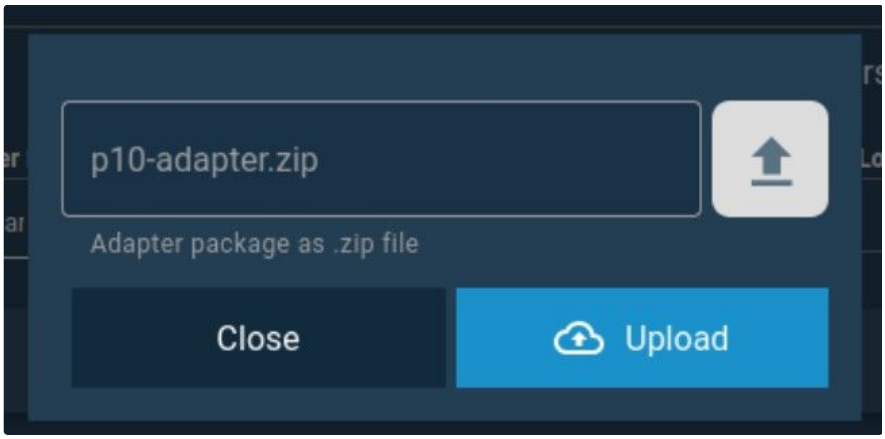
Note: The zip file name should be same as the json/.tar.gz files.

- **Importing the Adapter:** Log into IF Manager, select the Connections menu, click Import Adapters, and select the zip file you created. Upload the file and wait for the success message.

Log into the application with valid username and password.



Select the Connections menu, click Import Adapters and select From Local Drive the zip file you've created in the previous steps.



Select your adapter and click Upload (wait for the success message).

Managing Connections

After importing the adapter, you can manage connections in IF Manager. Connections are running instances of adapters with specific configurations. You can create, edit, and delete connections as needed.

Monitoring and Troubleshooting

IF Manager provides tools for monitoring the status of connections and troubleshooting any issues that arise. You can check connection status, view logs, and access health and status endpoints to ensure connections are running smoothly.

Kubernetes Dashboard

The Kubernetes dashboard is a web-based user interface that provides a graphical interface for managing Kubernetes clusters. It allows users to view and manage various aspects of their Kubernetes cluster, such as deployments, pods, services, and more. Accessing the Kubernetes dashboard is essential for Kubernetes cluster administrators and developers to monitor and manage their applications effectively.

Accessing the Kubernetes dashboard provides an intuitive interface for managing your Kubernetes cluster. By following the steps outlined above, you can easily access the dashboard and leverage its features to monitor and manage your Kubernetes applications effectively.

Prerequisites

Before accessing the Kubernetes dashboard, ensure that you have the following prerequisites:

1. A running Kubernetes cluster.
2. kubectl command-line tool installed and configured to connect to your Kubernetes cluster.

Steps to Access the Kubernetes Dashboard

Follow these steps to access the Kubernetes dashboard:

1. Retrieve the Dashboard URL:

Use the following command to retrieve the URL for accessing the Kubernetes dashboard:

kubectl cluster-info | grep dashboard

2. This command will output the URL for accessing the Kubernetes dashboard.shell

3. Create a Dashboard Service Account:

If you haven't already created a service account for the dashboard, you can create one using the following command: kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.4.0/aio/deploy/recommended.yaml

4. Create a Cluster Role Binding:

To grant the necessary permissions for the dashboard service account, create a cluster role binding using the following command: kubectl create clusterrolebinding dashboard-admin-sa --clusterrole=cluster-admin --serviceaccount=kubernetes-dashboard:dashboard-admin-sa

5. Access the Dashboard:

Open the URL retrieved in step 1 in your web browser. You will be prompted to log in. Select "Token" as the authentication method and paste the token for the dashboard service account created in step

6. View and Manage your Kubernetes Cluster:

Once logged in, you will be able to view and manage various aspects of your Kubernetes cluster using the dashboard interface. You can view information about your cluster, manage deployments, pods, services, and more.

Kubernetes Cluster Issues

If a Kubernetes cluster experiences downtime due to resource issues, it's important to identify and address the cause systematically.
Follow these steps to recover issues and prevent future occurrences:

Identify the Root Cause

1. Ensure that all nodes are available and functioning by running: `kubectl get nodes`
   Identify nodes that are in a NotReady state or Unknown.
2. Look for pods in Pending, CrashLoopBackOff, or OOMKilled states: `kubectl get pods --all-namespaces`
3. Use monitoring tools (e.g., kubectl top or Prometheus) to check CPU, memory, and disk utilization:
   `kubectl top nodes`
   `kubectl top pods`
4. **Check Kubernetes Events**
   Kubernetes events are a useful tool for troubleshooting resource issues because they provide detailed, real-time information about what is happening with Kubernetes objects (like Pods, Nodes, Services, etc.). Events track changes in the state of resources and the actions Kubernetes takes in response to those changes. Here's how Kubernetes events help in troubleshooting:
   - **Understanding Resource Lifecycle Events**
     Kubernetes creates events whenever something important happens to a resource, such as a Pod failing to start or a Node becoming unavailable. These events provide insights into what happened before a resource issue occurred, helping you trace the cause.
     Examples of Events:
     - Pod scheduled on a node: Successfully assigned pod-name to node-name
     - Container starting/stopping: Started container / Killing container
     - Image pull issues: Failed to pull image
   - **Diagnosing Errors in Pods**

     Events help you see why a Pod might be in a non-running state, such as:
     - **ImagePullBackOff**: Shows that Kubernetes failed to pull the container image due to incorrect image tags or registry issues.
     - **CrashLoopBackOff**: Indicates that a container repeatedly crashed after starting.
     - **FailedScheduling**: Suggests a lack of resources (CPU, memory) or taints on the Node, preventing Pods from being scheduled.
   - **Network and Connectivity Issues**

     Events can reveal issues with Services, Endpoints, or network policies. For instance:
     - **Failed to update endpoints**: May indicate that a service cannot find any matching Pods.
     - **Pod sandbox changed**: Shows connectivity issues between the Node and the Pod.
   - **Resource Constraints**

     Kubernetes events help troubleshoot when resource limits and requests are misconfigured:
     - **OOMKilled**: The container was killed because it exceeded its memory limit.
     - **FailedScheduling**: The Pod cannot be scheduled due to insufficient CPU or memory on the available Nodes.
   - **Node and Cluster-Level Issues**

     Events also reflect problems at the node or cluster level, like:

- **NodeNotReady**: A Node is marked as not ready, which could be due to networking or resource issues.
- **DiskPressure**: Indicates disk space is running low on a Node.
- **NodeHasNoDiskPressure**: The Node recovered from disk pressure.

5. **Disk/Storage Pressure**:
   - Check if there is storage pressure on nodes or persistent volumes.
   - Inspect the node's disk usage: `kubectl describe node <node-name> | grep -i "disk"`
   - Use df -h or similar tools on the node itself to check storage space.
6. **Pod OOMKills**:
   Pods may be getting OOMKilled if they consume more memory than requested or available. Check for OOMKilled events: `kubectl get events --all-namespaces | grep -i "oomkilled"`
7. **Check system logs:**
   Review logs on Kubernetes control plane components (`like kube-apiserver, kube-controller-manager, etcd`) and worker nodes for errors.
   Logs can be accessed using `journalctl` or by running below to investigate issues related to resource exhaustion.
   - `kubectl` logs on the problematic pods or
   - access node logs (`/var/log/syslog or /var/log/kubelet.log`)

Free up Resources

1. Use the kubectl drain command to temporarily evict non-essential pods from heavily loaded nodes, allowing critical services to recover:
   `kubectl drain <node-name> --ignore-daemonsets --force`
2. If some workloads are resource-heavy, temporarily scale them down to free up CPU/memory:
   `kubectl scale deployment <deployment-name> --replicas=<desired-number>`
3. For critical pods, ensure they have sufficient resource requests and limits.
   a. Update deployments to increase resource requests or limits as needed.
4. If a node is running out of resources, you may need to drain it:
   `kubectl drain <node-name> --ignore-daemonsets --delete-emptydir-data`

Allocate More Resources

1. If the overall cluster resource is inadequate, consider scaling your cluster by adding more nodes (manually or via autoscaling if configured):
   `kubectl apply -f <new-node-config>.yaml`
2. Modify resource requests/limits for pods to prevent overcommitment.

```
resources:
  requests:
    memory: "64Mi"
    cpu: "250m"
  limits:
    memory: "128Mi"
    cpu: "500m"
```

Review Cluster Resource Allocation

1. Ensure that resource requests and limits set for containers are realistic. Overcommitment can lead to resource starvation and node instability.
2. Set resource quotas to limit excessive resource consumption.

```
apiVersion: v1
kind: ResourceQuota
```

```
metadata:
  name: compute-resources
  namespace: <namespace>
spec:
  hard:
    requests.cpu: "1"
    requests.memory: "1Gi"
    limits.cpu: "2"
    limits.memory: "2Gi"
```

Use Horizontal Pod Autoscaling (HPA)

Set up Horizontal Pod Autoscaling (HPA) to automatically scale your application based on CPU/memory utilization.
`kubectl autoscale deployment <deployment-name> --cpu-percent=80 --min=2 --max=10`

Monitor the Cluster

Tools like Prometheus and Grafana can help you monitor CPU, memory, and storage usage in real-time. Set up alerts to catch resource issues early before they cause a full cluster failure.

Use Kubernetes events to monitor for OOM kills, failed scheduling, or node pressure events.

> **Note**: Kubernetes and Grafana are already setup on most of our SmartHive clusters. You can refer and monitor the dashboards)

1. **Set Up Prometheus and Grafana on Kubernetes:**
   Deploy Prometheus to collect metrics and Grafana to visualize them. The easiest way is to use the Prometheus Operator or kube-prometheus stack which includes Prometheus, Grafana, Alertmanager, and some pre-configured dashboards for Kubernetes.
2. **Configure Prometheus to Scrape Kubernetes Metrics:**
   By default, Prometheus will scrape Kubernetes metrics from the following:
   - **Nodes:** CPU, memory, disk, and other resource usage statistics.
   - **Pods:** CPU and memory usage, pod statuses, and more.
     The scrape configuration for Kubernetes is typically included in `kube-prometheus-stack` out-of-the-box, but you can customize it if necessary.
     Ensure Prometheus is scraping these key metrics:
     - `kube_node_status_condition` (to check node conditions like memory pressure, disk pressure, etc.)
     - `kube_pod_status_phase` (to check pod statuses like Running, Failed, Pending)
     - `node_cpu`, `node_memory_MemAvailable_bytes` (for node resource monitoring)
     - `container_memory_working_set_bytes`, `container_cpu_usage_seconds_total` (for pod/container-level resource monitoring)
3. **Set Up Grafana Dashboards:**
   Grafana comes with several built-in dashboards for Kubernetes monitoring, but you can also import dashboards from the Grafana Dashboard website.
   The recommended Dashboards are,
   - Kubernetes Cluster Monitoring: Dashboard
   - Kubernetes Pod Resources: Dashboard
     To import dashboards:
     i. Go to Grafana UI -> Dashboards -> Manage -> Import.
     ii. Enter the Dashboard ID (e.g., `315`) and click "Load".
     iii. Select the Prometheus data source and save.

4. **Set Alerts in Prometheus for Node and Pod Resource Issues:**
   Set up alerts in Prometheus Alertmanager for node and pod resource issues. Some key alert rules include:
   a. **For Node Resource Monitoring:**
      i. Node Out of Memory: Trigger an alert if a node's available memory falls below a certain threshold.

```
- alert: NodeMemoryLow
  expr: node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes * 100 < 10
  for: 2m
  labels:
    severity: warning
  annotations:
    summary: "Node Memory Low"
description: "The node has less than 10% available memory (instance {{ $labels.instance }})"
```

      ii. Node CPU Saturation: Alert when a node's CPU usage is critically high.

```
- alert: NodeCPUHigh
  expr: (1 - avg by (instance) (rate(node_cpu_seconds_total{mode="idle"}[5m]))) * 100 > 90
  for: 2m
  labels:
    severity: critical
  annotations:
    summary: "Node CPU Saturation"
    description: "CPU usage on node (instance {{ $labels.instance }}) is above 90%"
```

   b. **For Pod Resource Monitoring:**
      i. Pod OOMKilled: Alert when a pod is killed due to an out-of-memory error.

```
- alert: PodOOMKilled
  expr: kube_pod_container_status_terminated_reason{reason="OOMKilled"} > 0
  for: 2m
  labels:
    severity: critical
  annotations:
    summary: "Pod OOMKilled"
    description: "Pod {{ $labels.pod }} in namespace {{ $labels.namespace }} is OOMKilled"
```

      ii. Pod CrashLoopBackOff: Alert if a pod enters a crash loop.

```
- alert: PodCrashLoopBackOff
  expr: kube_pod_container_status_waiting_reason{reason="CrashLoopBackOff"} > 0
  for: 2m
  labels:
    severity: critical
  annotations:
    summary: "Pod CrashLoopBackOff"
    description: "Pod {{ $labels.pod }} in namespace {{ $labels.namespace }} is CrashLoopBackOff"
```

5. **Visualizing and Monitoring in Grafana:**
   - In the Grafana dashboard, you'll see real-time data on the state of nodes, pods, and overall cluster health.
   - Use Grafana Alerts for custom visual alerts or integrate Alertmanager to trigger email or Slack notifications when nodes or pods go down or resources are exhausted.
6. **Troubleshooting & Optimization:**
   a. Use Grafana panels to drill down into specific nodes or pods when resource issues are flagged.
   b. Combine metrics like CPU/memory usage with pod status to identify bottlenecks or problematic containers.
   c. Set thresholds in Prometheus for alerting based on your infrastructure’s capacity and load expectations.

Review Storage Issues

- Check storage availability. If Persistent Volume Claims (PVCs) or volumes are exhausted, consider expanding them.
  `kubectl describe pvc <pvc-name>`
  `kubectl edit pvc <pvc-name>`
- Add storage resource requests and limits to prevent pods from consuming excessive storage.

Restart Critical Components

If the problem is related to kubelet or critical system services, restarting them might help.

`systemctl restart kubelet`

Review Networking Resources

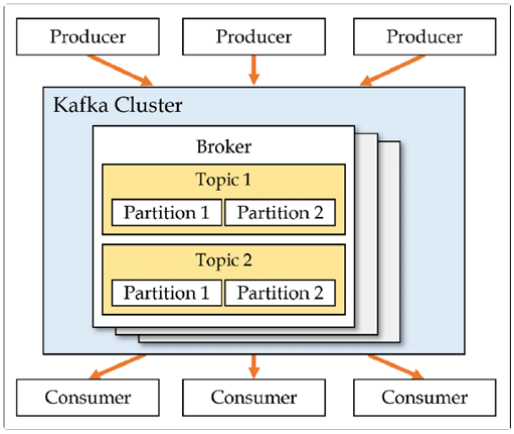
Check network policies and logs for any issues with connectivity, DNS resolution, or high latency affecting cluster performance.

Take Preventative Measures

- Enable Cluster Autoscaler to dynamically add nodes when the existing ones run out of resources.
- Use taints and tolerations to protect critical workloads by ensuring they only run on nodes with adequate resources.
- Define Quality of Service (QoS) classes for pods (Guaranteed, Burstable, BestEffort) to prioritize resources for critical workloads.

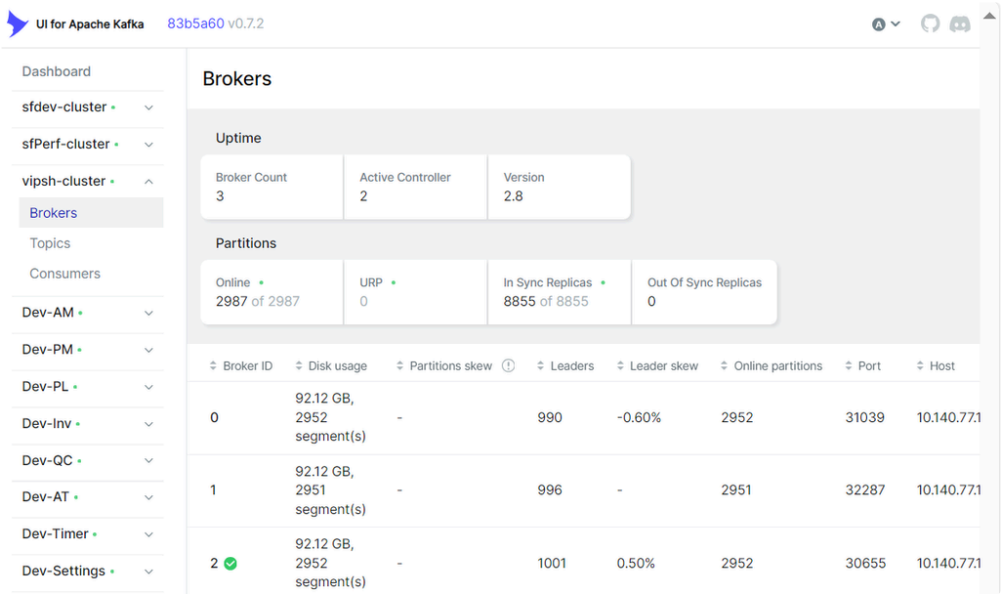
Apache Kafka

Apache Kafka is a distributed streaming platform designed for building real-time data pipelines and streaming applications. It is a publish-subscribe-based message system. Producers publish messages, and consumers consume or pull that data. It is used to handle large volumes of data efficiently and in a fault-tolerant manner.

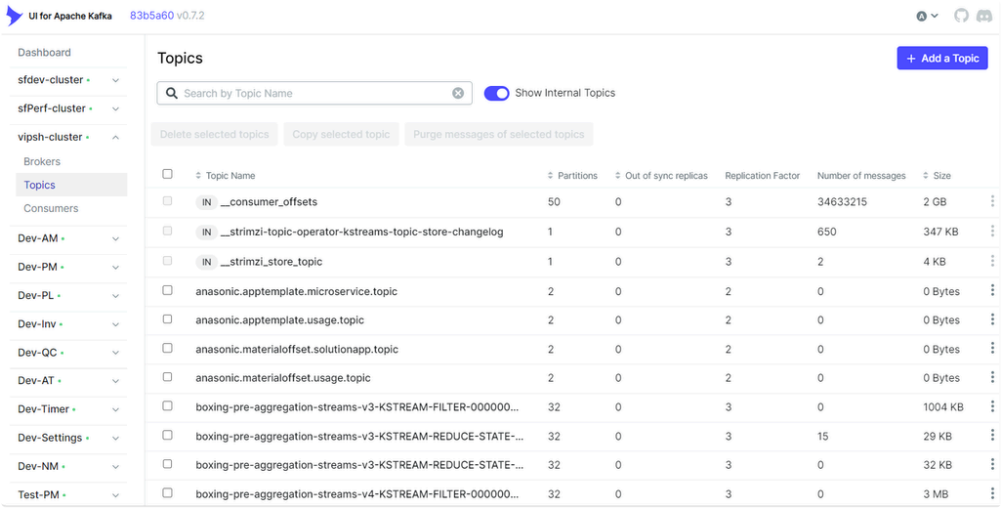


The following are the key components of Apache Kafka:

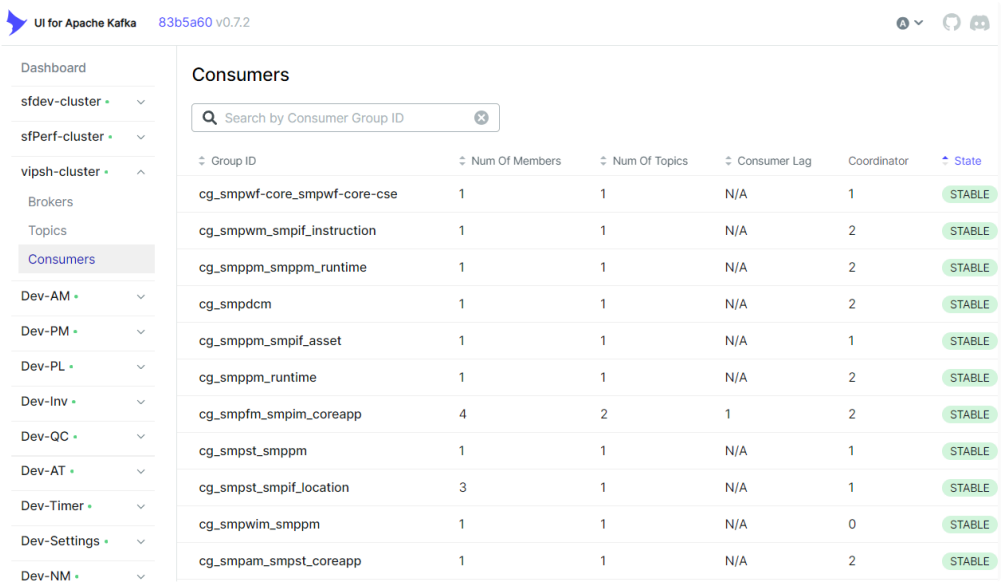
- **Kafka Broker**: A Kafka broker is a server that stores data and serves client requests. In a Kafka cluster, multiple brokers work together to manage the storage and retrieval of messages. Each broker handles a subset of partitions and manages the data flow, ensuring high availability and scalability.



- **Topics**: A topic in Kafka is a category or feed name to which messages are published. Topics are partitioned, meaning that each topic can be divided into multiple partitions, allowing for parallel processing and better performance. Each partition is an ordered, immutable sequence of messages, and Kafka retains messages for a configurable amount of time.



- **Producers**: Kafka producers send messages to one or more topics. Data is sent to the Kafka cluster. When a Kafka producer sends a message to Kafka, the broker receives it and appends it to one partition. Producers have the option of publishing messages to a certain division.
- **Consumers**: Consumers are applications that subscribe to topics and process the messages. They can read messages from one or more topics and can be part of a consumer group, where each consumer in the group reads from a subset of the partitions. This allows for load balancing and fault tolerance, as multiple consumers can work together to process data efficiently.



- **ZooKeeper**: Kafka uses ZooKeeper to manage and organize everything. It is used by Kafka to determine which broker is the leader for each partition. When a leader broker goes offline, ZooKeeper will detect the failure and trigger a leader election process. Each partition has only one leader broker responsible for read and write operations, while the other brokers act as followers and replicate data in case of failure.

  Additionally, Kafka uses ZooKeeper for Configuration Management. It stores important settings and information about the Kafka cluster, ensuring that all the brokers have the same understanding of how things work.

Troubleshooting Kafka Issues

Follow these steps to identify and resolve common problems with Kafka:

1. **Check Broker Status**
   a. Check the Kafka broker logs (server.log) for any errors or warnings. Logs are usually located in the log's directory defined in your Kafka configuration.

b. Use the Kafka CLI tool to check if all brokers are running:
`kafka-broker-api-versions --bootstrap-server <broker_address>`

2. **Verify Zookeeper Connection**
   a. Ensure that Kafka is correctly connected to Zookeeper. Check Zookeeper logs for any issues.
   b. Use the Zookeeper CLI to check the status:
   `zkCli.sh -server <zookeeper_address> status`
3. **Check Topic Configuration**
   a. Verify that the topic exists:
   `kafka-topics --list --bootstrap-server <broker_address>`
   b. Check the topic's configuration and partitions:
   `kafka-topics --describe --topic <topic_name> --bootstrap-server <broker_address>`
4. **Producer Issues**
   a. Check if the producer is sending messages successfully. Look for exceptions or errors in the producer logs.
   b. Ensure that the producer's configuration for retries is set appropriately.
   c. Acknowledge Settings: Adjust acks to see if the issue resolves. For testing, set it to all.
5. **Consumer Issues**
   a. Check if the consumer is committing offsets correctly. Use:
   `kafka-consumer-groups --bootstrap-server <broker_address> --describe --group <consumer_group>`
   b. Monitor consumer lag using tools like kafka-consumer-groups.
   c. Ensure that the consumer configurations (like group.id, auto.offset.reset) are set correctly.
6. **Network Issues**
   a. Check network connectivity between producers, brokers, and consumers.
   b. Use tools like ping and telnet to verify connectivity on the necessary ports (default Kafka port is 9092).
7. **Resource Utilization**
   a. Monitor CPU, memory, and disk I/O on your Kafka brokers.
   b. Ensure there is enough disk space and check if the log segments are being deleted as per retention policies.
8. **Replication Issues**
   a. If replication is failing, check the replication factor of the topic and the state of replicas.
   b. Use the topic description command to look at the ISR (In-Sync Replica) list.
9. **Using Monitoring Tools**
   Use Kafka monitoring tools (like Kafka Manager, Confluent Control Center, Prometheus with Grafana) to get insights into performance metrics and health.
10. **Version Compatibility**
    Ensure that the Kafka client and broker versions are compatible. Mismatched versions can lead to unexpected behavior.

Grafana Dashboards

The Grafana dashboard is a visual representation of data and metrics, providing a consolidated view of various metrics and information in a single, customizable interface. Grafana is an open-source analytics and monitoring platform that integrates with various data sources, such as time-series databases, to create dynamic and interactive dashboards.

The Grafana dashboard has a collection of panels displayed in a grid. Each panel displays data in visual form, such as graphs, charts, and other visualizations. These panels are created using components that transform raw data from a data source into visualizations.

By visualizing the metrics in real time, you can gain insights into the overall health and performance of the cluster, identify potential bottlenecks or issues, and make informed decisions to optimize cluster operations. With intuitive graphs and charts, the dashboard enables proactive monitoring, troubleshooting, and capacity planning, ensuring the smooth and efficient functioning of Kubernetes clusters. For the list of Grafana alerts, refer List of Grafana Alerts.

Prerequisites

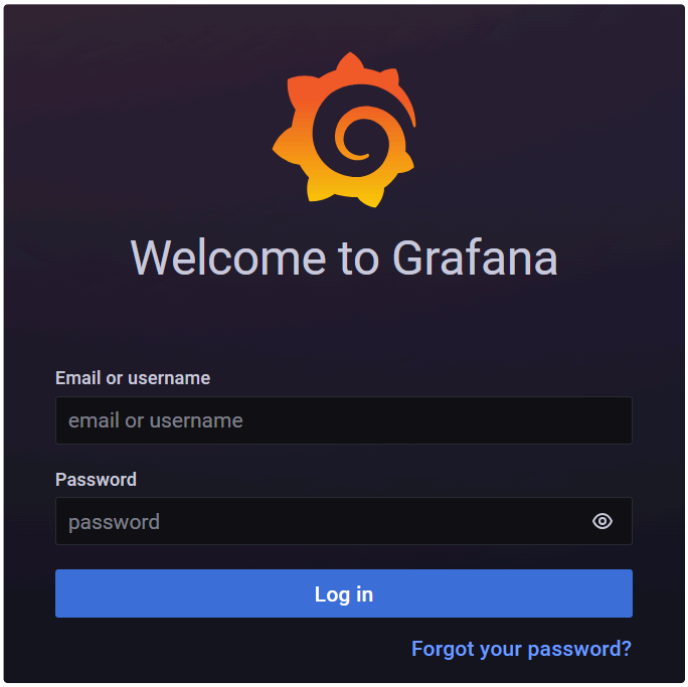
Before accessing the Grafana dashboard, ensure that you have the following prerequisites:

1. A running Grafana instance.
2. Access to the Zscaler Client Connector.
3. Access to the Grafana dashboard: https://<customer domain name>/grafana (e.g. https://vip-sh.panasonicfa.com/grafana).
4. Credentials (username and password) for logging in to Grafana.

Log In

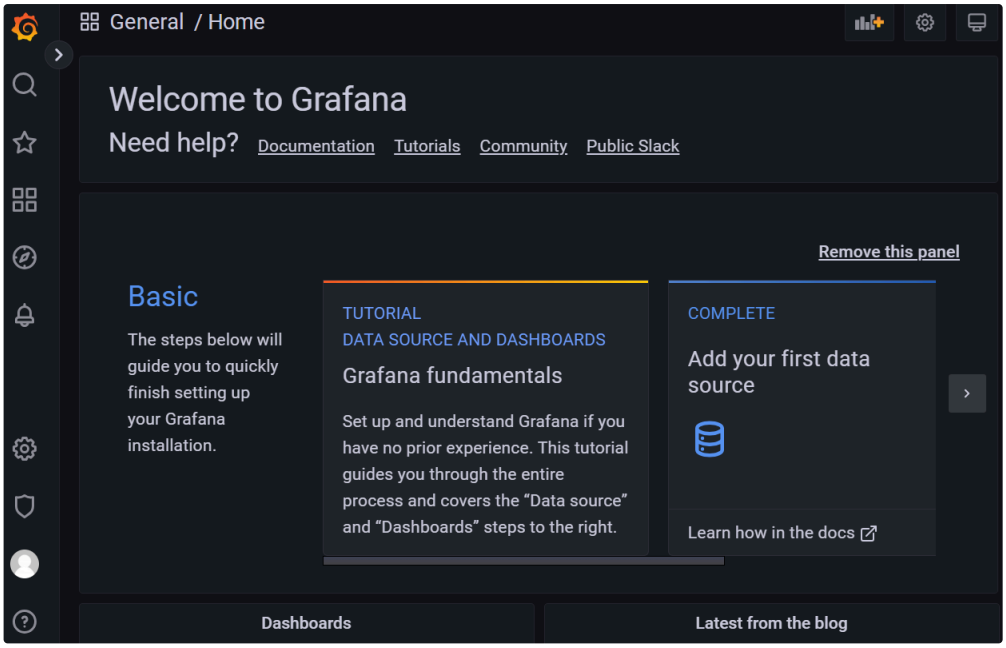
Follow these steps to login to the Grafana dashboard:

1. Open a web browser and enter the URL: https://<customer domain name>/grafana (e.g. https://vip-sh.panasonicfa.com/grafana) to access the Grafana.
   The Grafana Login page appears.
2. Enter username and password in their respective fields.



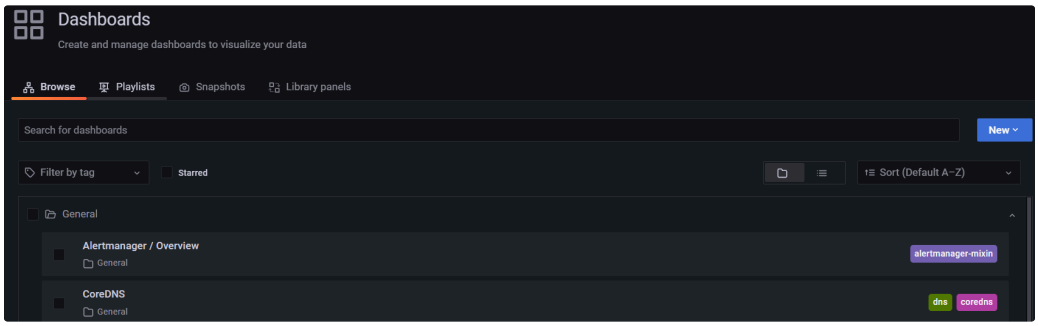
Note: If you have forgotten your password, click the **Forgot your password?** link and follow the instructions to reset your password.

3. Click **Log in**.
   The Grafana Home page appears when you logged in.



Standard or Preconfigured Grafana Dashboards

To access the standard or preconfigured Grafana dashboards, go to **Dashboards** > **Browse** from the sidebar menu and search for the dashboard.



The following are the list of preconfigured Grafana dashboards:

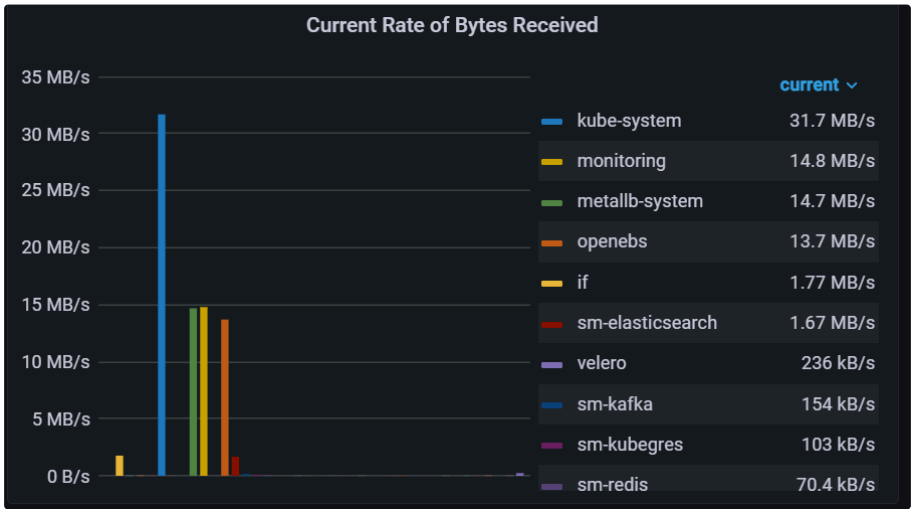
Kubernetes / Networking / Cluster Dashboard

The Kubernetes Networking/Cluster Grafana Dashboard offers a centralized platform for monitoring the networking performance and overall health of a Kubernetes cluster. It provides insights into various networking components such as ingress controllers/CNI and enabling administrators to track metrics like throughput, latency, and error rates. Additionally, the dashboard facilitates observation of cluster-wide metrics such as node status, pod distribution, and resource utilization, empowering operators to detect anomalies, troubleshoot issues, and scale resources effectively. Overall, it serves as a vital tool for ensuring the reliability and efficiency of containerized workloads in Kubernetes environments.

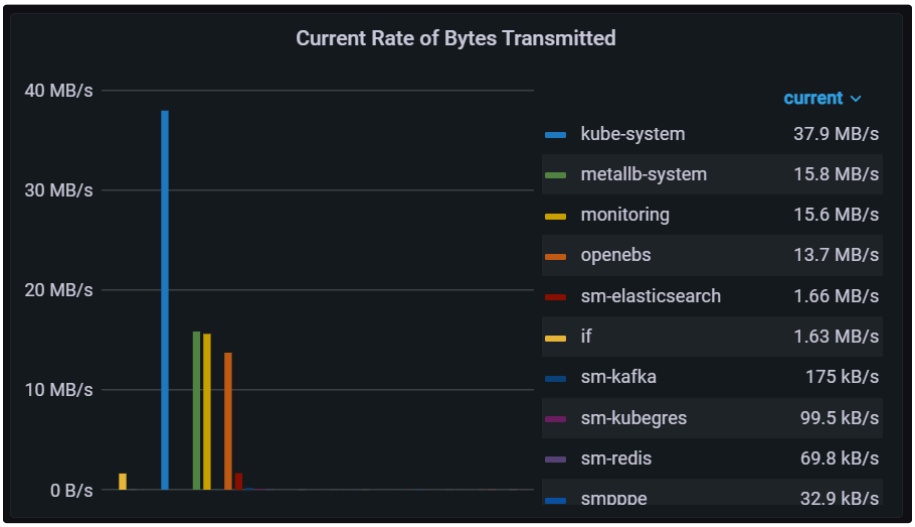
By analyzing the metrics shown in the panels, operators can identify trends, troubleshoot connectivity issues and ensure efficient data reception or transmission to support the smooth operation of Kubernetes workloads.

The following are the panels of the Kubernetes / Networking / Cluster Dashboard:

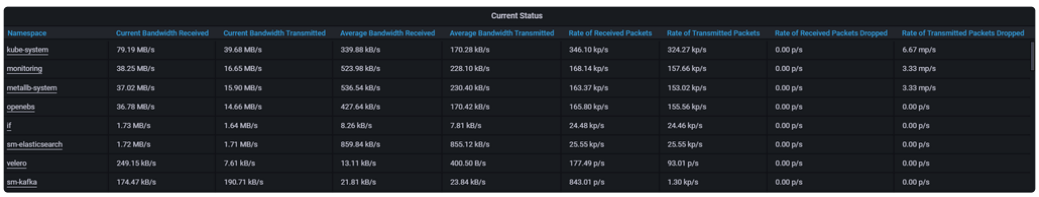
Current Rate of Bytes Received: This panel provides a real-time summary of the amount of incoming network traffic within the Kubernetes cluster. It offers administrators insights into the volume of data being received by the cluster, facilitating the monitoring of network performance and the identification of potential issues such as congestion or anomalies. This metric enables operators to promptly address connectivity issues, optimize network configurations, and ensure the efficient operation of Kubernetes workloads.



Current Rate of Bytes Transmitted: This panel provides a real-time summary of the volume of outgoing network traffic within the Kubernetes cluster. It offers administrators insights into the amount of data being transmitted from the cluster to external destinations, facilitating the monitoring of network performance and the detection of potential issues such as congestion or anomalies.

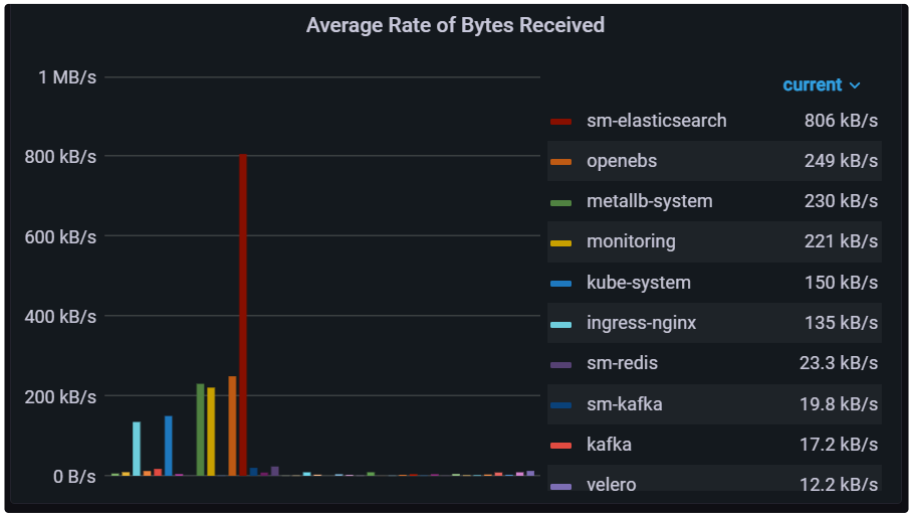


Current Status: This panel provides a concise summary of the overall health and status of the networking components within the Kubernetes cluster. It typically displays key indicators such as network connectivity, service availability, and node communication status. This panel serves as a quick reference for administrators to assess the general well-being of the cluster's networking infrastructure at a glance.

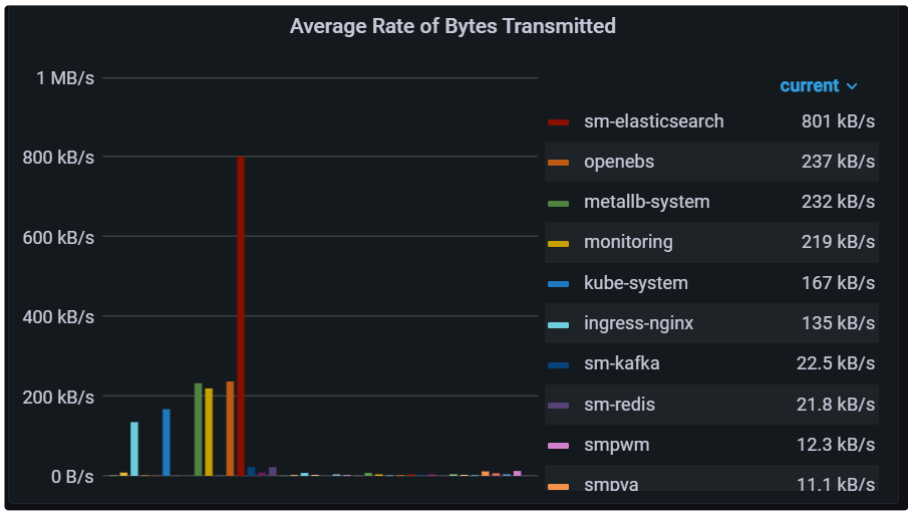

Current Status

Namespace	Current Bandwidth Received	Current Bandwidth Transmitted	Average Bandwidth Received	Average Bandwidth Transmitted	Rate of Received Packets	Rate of Transmitted Packets	Rate of Received Packets Dropped	Rate of Transmitted Packets Dropped
kube-system	79.19 MB/s	39.68 MB/s	339.88 kB/s	170.28 kB/s	346.10 kp/s	324.27 kp/s	0.00 p/s	6.67 mp/s
monitoring	38.25 MB/s	16.65 MB/s	523.98 kB/s	228.10 kB/s	168.14 kp/s	157.66 kp/s	0.00 p/s	3.33 mp/s
metallb-system	37.02 MB/s	15.90 MB/s	536.54 kB/s	230.40 kB/s	163.37 kp/s	153.02 kp/s	0.00 p/s	3.33 mp/s
openebs	36.78 MB/s	14.66 MB/s	427.64 kB/s	170.42 kB/s	165.80 kp/s	155.56 kp/s	0.00 p/s	0.00 p/s
if	1.73 MB/s	1.64 MB/s	8.26 kB/s	7.81 kB/s	24.48 kp/s	24.46 kp/s	0.00 p/s	0.00 p/s
sm-elasticsearch	1.72 MB/s	1.71 MB/s	859.84 kB/s	855.12 kB/s	25.55 kp/s	25.55 kp/s	0.00 p/s	0.00 p/s
velero	249.15 kB/s	7.61 kB/s	13.11 kB/s	400.50 B/s	177.49 p/s	93.01 p/s	0.00 p/s	0.00 p/s
sm-kafka	174.47 kB/s	190.71 kB/s	21.81 kB/s	23.84 kB/s	843.01 p/s	1.30 kp/s	0.00 p/s	0.00 p/s

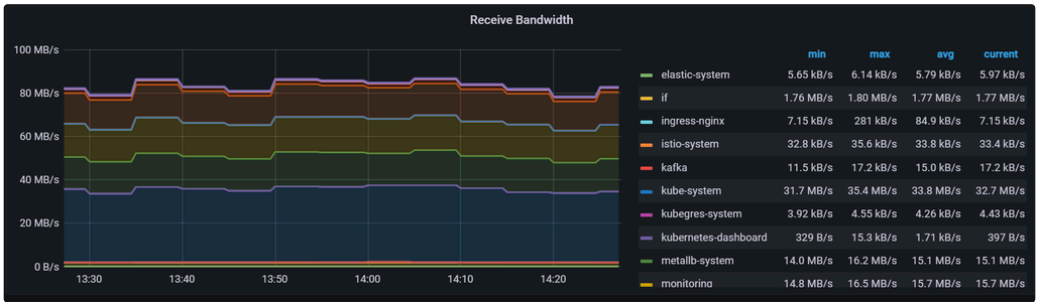
Average Rate of Bytes Received: This panel provides a concise summary of the average volume of incoming network traffic within the Kubernetes cluster over a specified time period. It offers administrators insights into the typical amount of data being received by the cluster, facilitating the monitoring of network performance trends and the identification of potential issues such as spikes or drops in traffic.



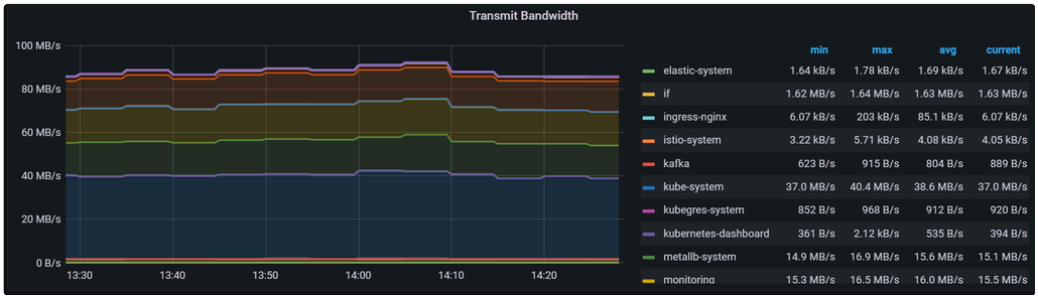
Average Rate of Bytes Transmitted: This panel offers a concise overview of the average volume of outgoing network traffic within the Kubernetes cluster over a specified timeframe. This panel provides administrators with insights into the typical amount of data transmitted from the cluster to external destinations, aiding in the monitoring of network performance trends and the identification of potential anomalies or inefficiencies.



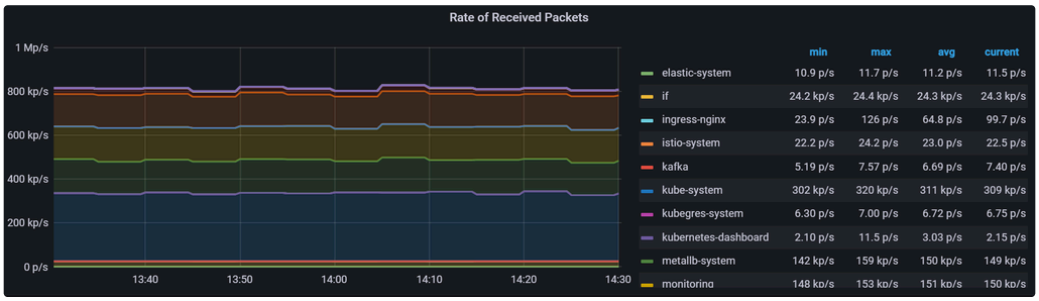
Receive Bandwidth: This panel provides a focused summary of the current bandwidth utilization dedicated to receiving network traffic within the Kubernetes cluster. This panel offers insights into the rate at which data is being received by the cluster from external sources, facilitating the monitoring of inbound network performance in real-time.



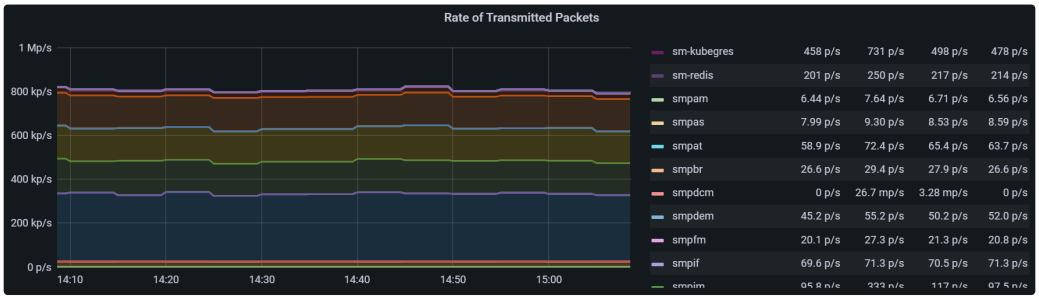
Transmit Bandwidth: This panel provides a concise overview of the current bandwidth utilization dedicated to transmitting network traffic from the Kubernetes cluster to external destinations. This panel offers insights into the rate at which data is being sent from the cluster to external sources, facilitating real-time monitoring of outbound network performance.



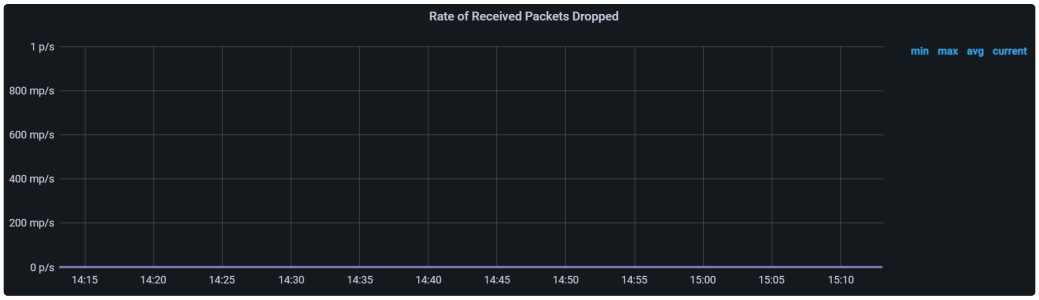
Rate of Received Packets: This panel provides a concise overview of the current rate at which packets are being received by the Kubernetes cluster from external sources. This metric offers insights into the volume of incoming network traffic in real-time, aiding administrators in monitoring network performance and detecting potential issues such as congestion or anomalies.



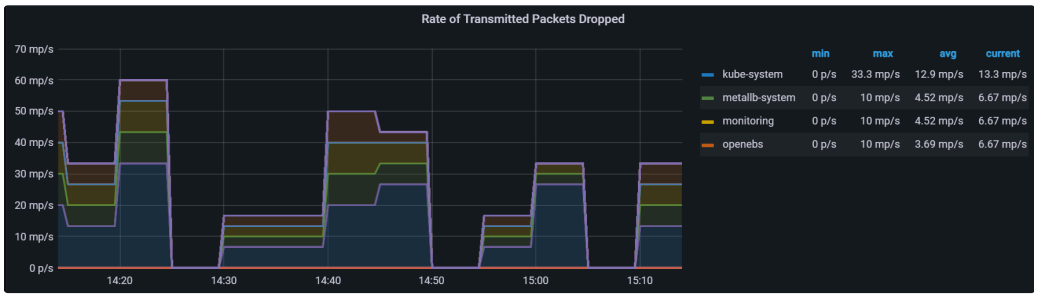
Rate of Transmitted Packets: This panel provides a concise overview of the current rate at which packets are being transmitted from the Kubernetes cluster to external destinations. This metric offers insights into the volume of outgoing network traffic in real-time, facilitating the monitoring of network performance and the detection of potential issues such as congestion or anomalies.



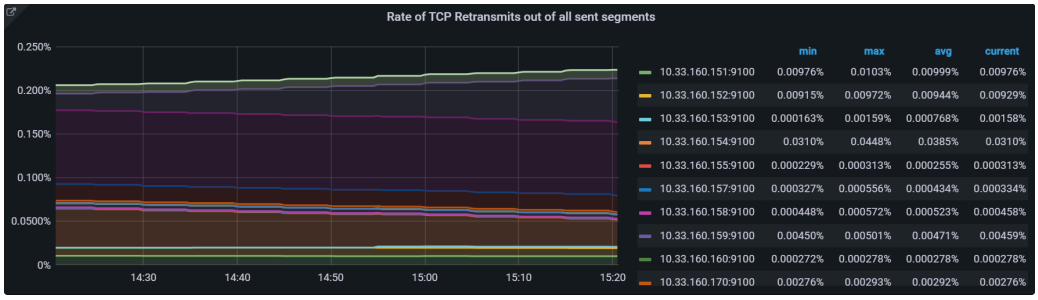
Rate of Received Packets Dropped: This panel provides a concise overview of the current rate at which incoming packets are being dropped within the Kubernetes cluster. This metric offers insights into potential network congestion or issues that may result in dropped packets, impacting data delivery and network performance.



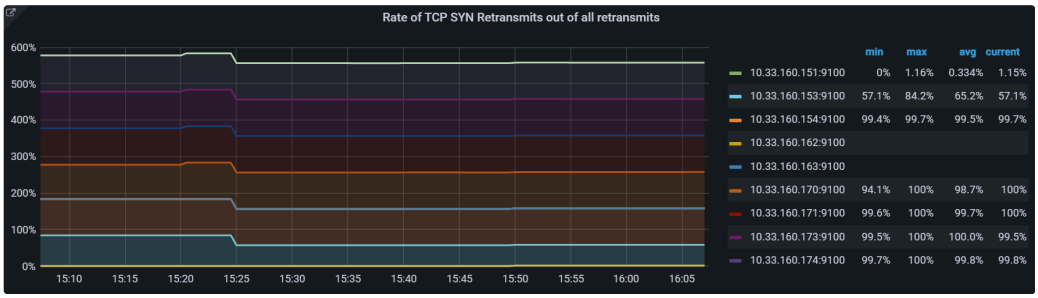
Rate of Transmitted Packets Dropped: This panel offers a concise summary of the current rate at which outgoing packets are being dropped within the Kubernetes cluster. This metric provides insights into potential network issues or congestion affecting packet transmission, which can impact data delivery and network performance. The focused view on dropped packet rates helps operators maintain optimal network reliability and responsiveness within the cluster.



Rate of TCP Retransmits out of all Sent Segments: This panel offers a concise summary of the frequency at which TCP segments are being retransmitted compared to the total number of segments sent within the Kubernetes cluster. This metric provides insights into potential network congestion, packet loss, or latency issues affecting TCP communication, which can impact the reliability and performance of network connections.



Rate of TCP SYN Retransmits out of all Retransmits: This panel provides a concise overview of the frequency at which TCP SYN (Synchronize) segments are being retransmitted compared to all retransmitted segments within the Kubernetes cluster. TCP SYN segments are the initial packets exchanged during the TCP three-way handshake, used to establish connections between network hosts. This metric offers insights into potential issues with connection establishment, such as network congestion, packet loss, or communication problems.



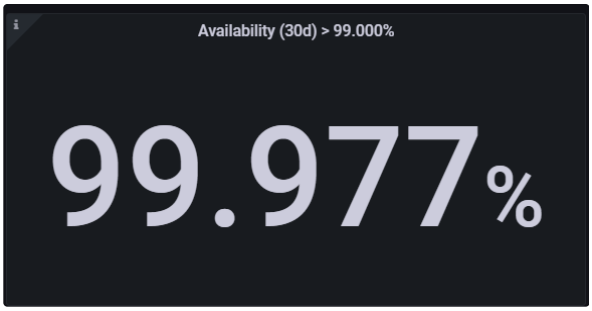
Kubernetes / API Server Dashboard

The Kubernetes API Server Grafana Dashboard serves as a vital tool for administrators to monitor and analyze the performance and health of the Kubernetes API server, which acts as the control plane component responsible for managing cluster operations. This dashboard provides a comprehensive view of key metrics such as request throughput, latency, error rates, and API server resource utilization.

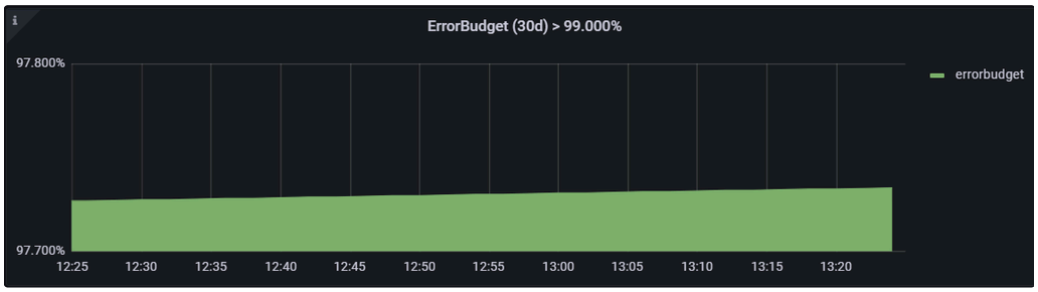
By monitoring the metrics of the panels, administrators can proactively address issues of the Kubernetes control plane and maintain the smooth operation of the cluster.

The following are the panels of the Kubernetes / API Server Dashboard:

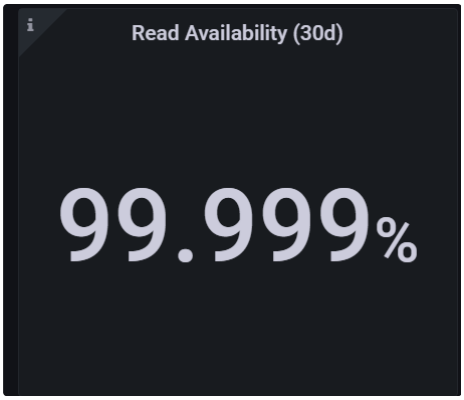
Availability: This panel provides a summary of the overall availability and uptime of the Kubernetes API server. It typically includes metrics related to the percentage of time the API server has been available within a specified time period, along with indicators of any downtime or disruptions. This panel helps administrators assess the reliability and resilience of the API server, enabling them to track service-level objectives (SLOs) and quickly identify any periods of unavailability or degradation in service.



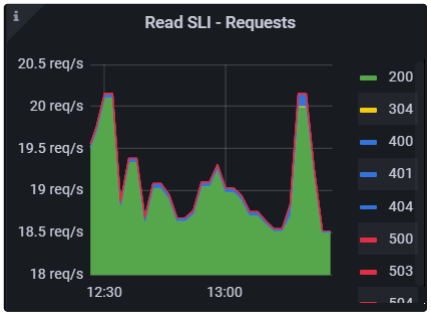
Error Budget: This panel provides a summary of the errors and failures experienced by the API server within a specified time frame. It typically includes metrics related to error rates, such as the percentage of failed requests or the number of HTTP error responses returned by the API server. This panel helps administrators track the health and reliability of the API server by monitoring deviations from acceptable error thresholds. Maintaining a healthy error budget is crucial for meeting service-level objectives (SLOs) and ensuring the smooth operation of Kubernetes clusters.



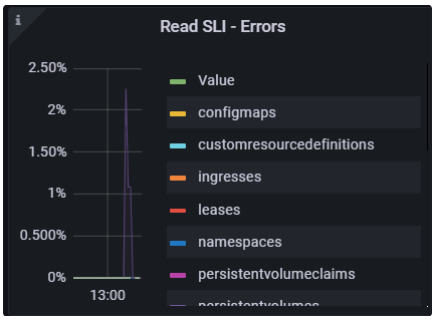
Read Availability: This panel provides insights into the availability and uptime specifically related to read operations performed by the API server. It typically includes metrics that quantify the percentage of successful read requests and the duration of read-related operations within a defined time period.



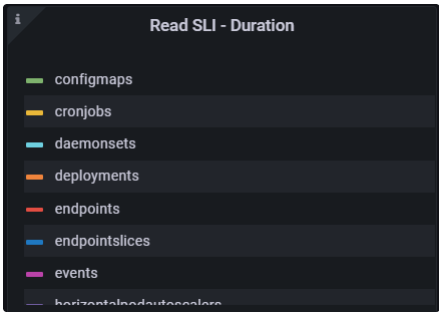
Read SLI Requests: This panel provides a summary of the service level indicators specifically related to read operations performed by the API server. It typically includes metrics that measure the success rate, latency, and throughput of read requests within a specified time period.



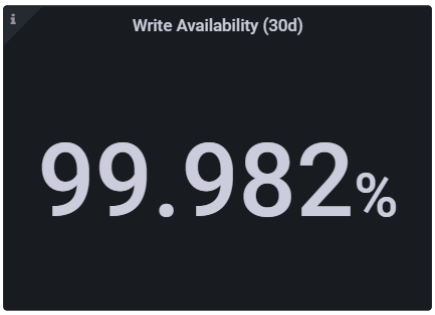
Read SLI - Errors: This panel offers insights into errors and failures specifically related to read operations performed by the API server. It typically presents metrics indicating the rate of errors encountered during read requests, such as the percentage of failed reads or the number of HTTP error responses returned by the API server. This panel helps administrators evaluate the reliability and effectiveness of read operations, allowing them to assess deviations from acceptable error thresholds and identify potential issues impacting the performance of the API server for read-intensive workloads.



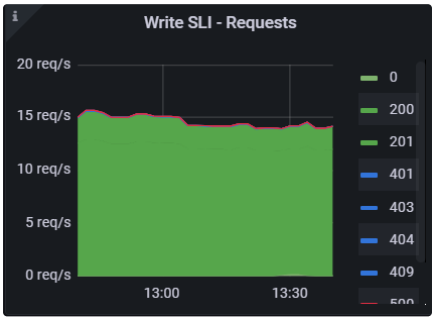
Read SLI - Duration: This panel provides a summary of the time taken for read operations performed by the API server. It typically includes metrics that measure the duration or latency of read requests, presenting information on the distribution of read operation durations over a specified time period. This panel helps administrators assess the responsiveness and efficiency of the API server for read-intensive workloads.



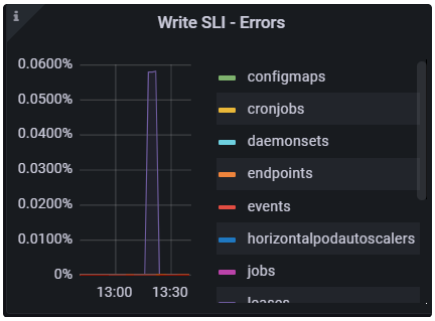
Write Availability: This panel provides insights into the availability and uptime specifically related to write operations performed by the API server. It typically includes metrics that quantify the percentage of successful write requests and the duration of write-related operations within a defined time period. This panel helps administrators assess the responsiveness and reliability of the API server for write-intensive tasks, such as creating or updating resources within the cluster.



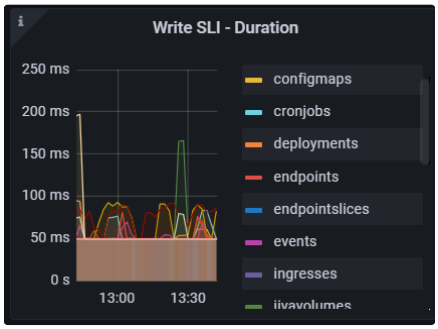
Write SLI Requests: This panel offers a summary of the service level indicators specifically related to write operations performed by the API server. It typically presents metrics indicating the success rate, latency, and throughput of write requests within a specified time period. This panel helps administrators gauge the performance and reliability of the API server for write-intensive tasks, such as creating or updating resources within the cluster.



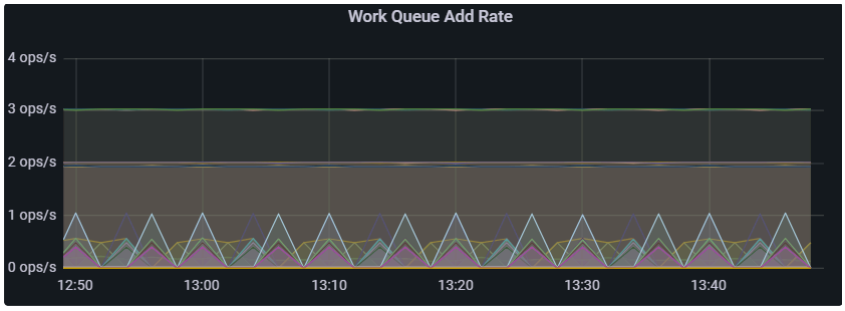
Write SLI Errors: This panel provides insights into errors and failures specifically related to write operations performed by the API server. It typically includes metrics indicating the rate of errors encountered during write requests, such as the percentage of failed writes or the number of HTTP error responses returned by the API server. This panel helps administrators evaluate the reliability and effectiveness of write operations, allowing them to assess deviations from acceptable error thresholds and identify potential issues impacting the performance of the API server for write-intensive tasks.



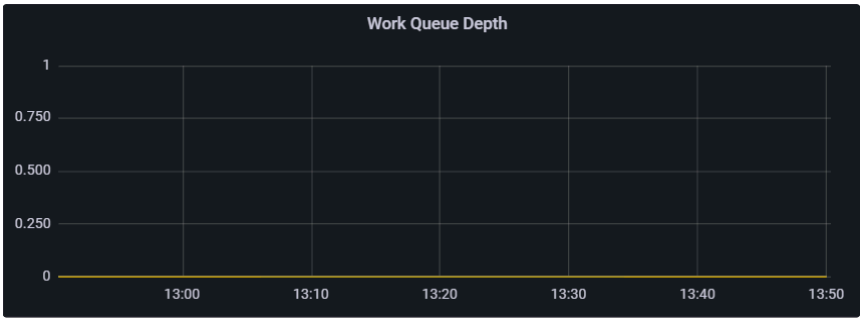
Write SLI Duration: This panel provides a summary of the time taken for write operations performed by the API server. It typically includes metrics that measure the duration or latency of write requests, presenting information on the distribution of write operation durations over a specified time period. This panel helps administrators assess the responsiveness and efficiency of the API server for write-intensive tasks, allowing them to identify potential performance bottlenecks or deviations from acceptable latency thresholds.



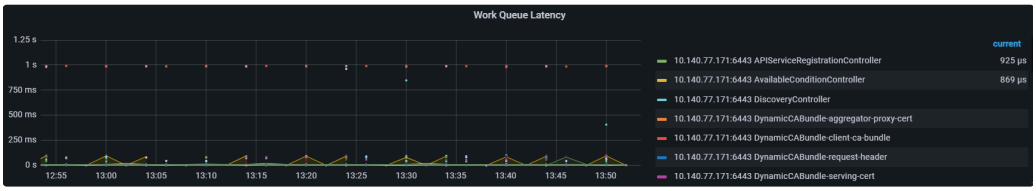
Work Queue Add Rate: This panel provides insights into the rate at which items are added to the work queue within the API server. It typically displays metrics indicating the frequency of additions to the work queue over time, allowing administrators to monitor the rate of incoming tasks or events that need to be processed by the API server. This panel helps administrators understand the workload and demand on the API server, enabling them to assess resource utilization, anticipate potential bottlenecks, and optimize performance.



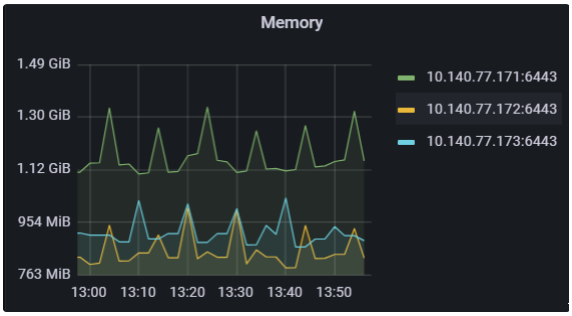
Work Queue Depth: This panel offers insights into the current backlog or depth of the work queue within the API server. It typically displays metrics representing the number of items currently waiting to be processed in the work queue over time. This panel helps administrators assess the workload and backlog of tasks within the API server, allowing them to monitor resource utilization, identify potential bottlenecks, and optimize performance.



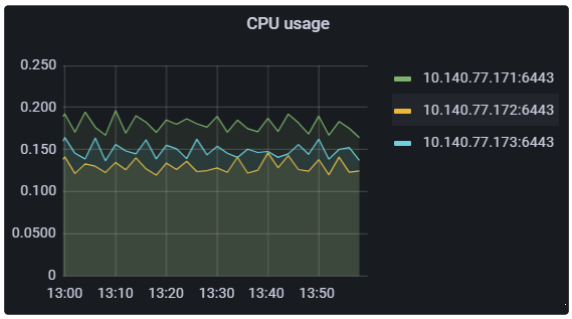
Work Queue Latency: This panel provides insights into the latency or delay experienced by items in the work queue within the API server. It typically displays metrics representing the time taken for items to be processed from the moment they are added to the work queue until they are handled or completed. This panel helps administrators assess the responsiveness and efficiency of task processing within the API server, allowing them to monitor performance, identify potential bottlenecks, and optimize resource allocation.



Memory: This panel offers insights into the memory utilization and performance of the API server. It typically includes metrics such as memory usage, available memory, and memory utilization trends over time. This panel helps administrators monitor the memory consumption of the API server, identify potential memory leaks or spikes, and ensure efficient resource management.

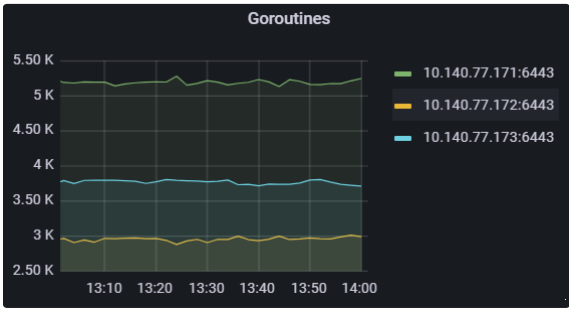


CPU Usage: This panel provides insights into the CPU utilization and performance of the API server. It typically includes metrics such as CPU usage, CPU load, and CPU utilization trends over time. This panel helps administrators monitor the CPU consumption of the API server, identify potential performance bottlenecks or spikes, and ensure efficient resource management.



Goroutines: This panel offers insights into the number of Goroutines actively running within the API server at any given time. Goroutines are lightweight threads managed by the Go programming language runtime, commonly used in

Kubernetes components like the API server for concurrent execution of tasks. This panel typically displays metrics representing the count of Goroutines over time, providing administrators with visibility into the concurrency level and workload handling capacity of the API server.



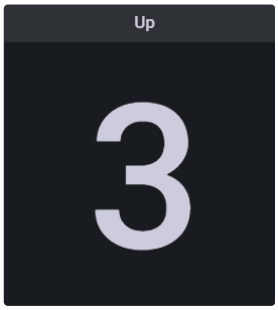
Kubernetes / Controller Manager Dashboard

The Kubernetes Controller Manager Grafana Dashboard serves as a pivotal tool for administrators to monitor and manage the Controller Manager component within Kubernetes clusters. As a core control plane component, the Controller Manager is responsible for managing various controllers that regulate the state of the cluster and perform tasks such as node management, pod scheduling, and replica set management. This dashboard provides real-time visibility into crucial metrics such as controller performance, event rates, and resource utilization. Through intuitive visualizations and charts, administrators can gain insights into the health and operation of controllers, detect anomalies, and troubleshoot issues efficiently.

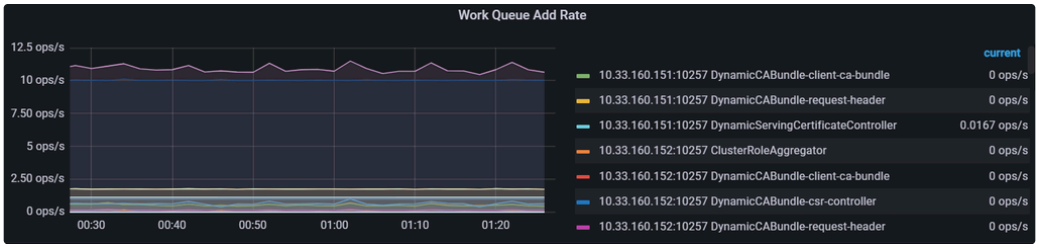
By leveraging this dashboard, administrators can ensure the stability, reliability, and optimal performance of Kubernetes clusters by proactively monitoring and managing the Controller Manager component.

The following are the panels of the Kubernetes / Controller Manager Dashboard:

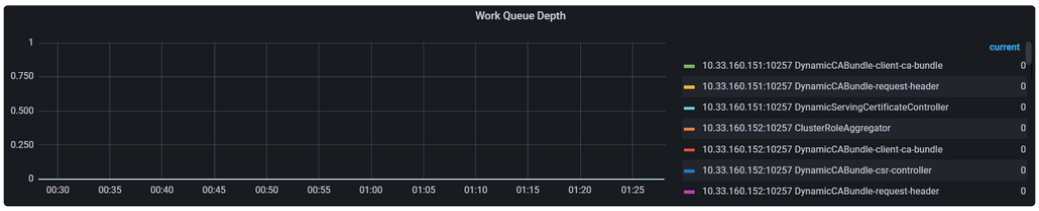
Up: This panel offers a concise summary of the availability and operational status of the Controller Manager component within the Kubernetes cluster. It typically displays metrics indicating whether the Controller Manager is running and accessible, providing administrators with a quick and easy way to assess the health of this critical control plane component.



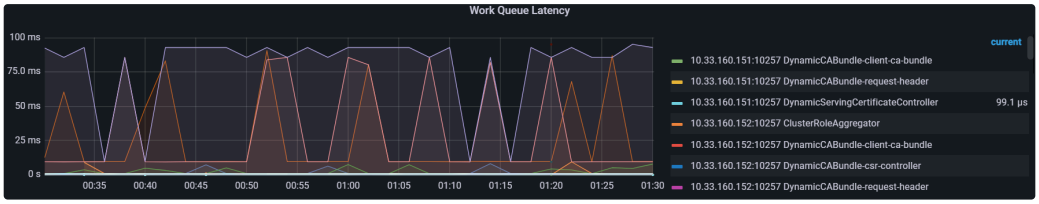
Work Queue Add Rate: This panel provides insights into the rate at which items are being added to the work queues managed by the Controller Manager. This panel typically displays metrics indicating the frequency of new tasks or events entering the work queues over time. Monitoring the add rate helps administrators understand the workload and activity levels of the Controller Manager, identify spikes in demand, and assess whether the system is keeping up with incoming tasks.



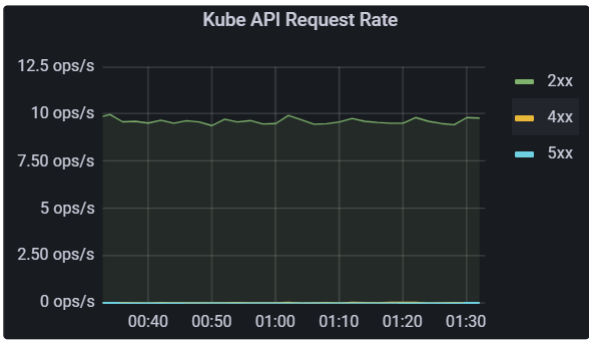
Work Queue Depth: This panel provides a visual representation of the current backlog in the work queues managed by the Controller Manager. It typically shows metrics that indicate the number of tasks or events waiting to be processed over time. This panel helps administrators assess the workload and efficiency of the Controller Manager by revealing how quickly it can handle queued tasks.



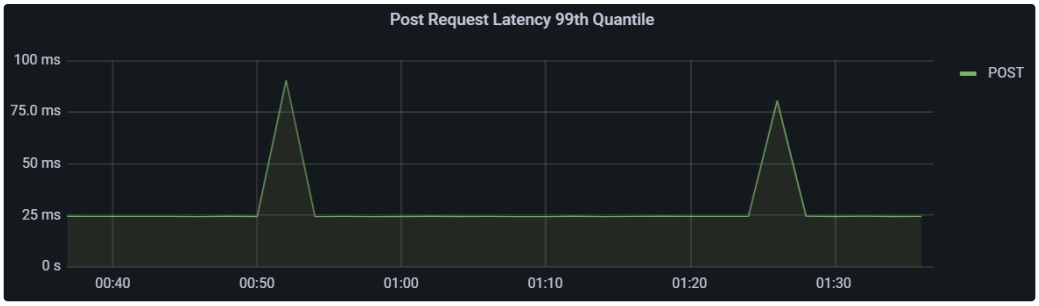
Work Queue Latency: This panel displays metrics related to the time tasks spend in the work queue before being processed by the Controller Manager. This panel typically shows the distribution and trends of task wait times, providing insights into the efficiency and responsiveness of the Controller Manager.



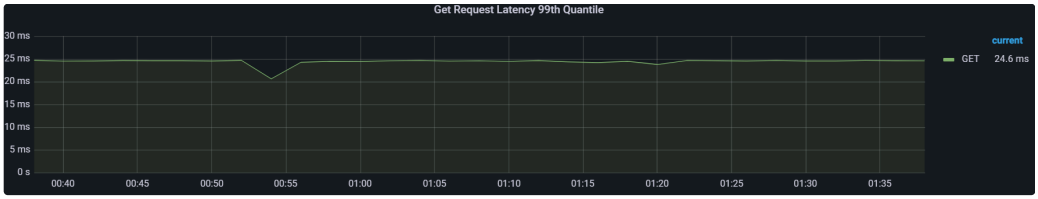
Kube API Request Rate: This panel provides metrics on the frequency of requests made by the Controller Manager to the Kubernetes API server. This panel typically displays the number of API requests over time, helping administrators monitor the interaction between the Controller Manager and the API server.



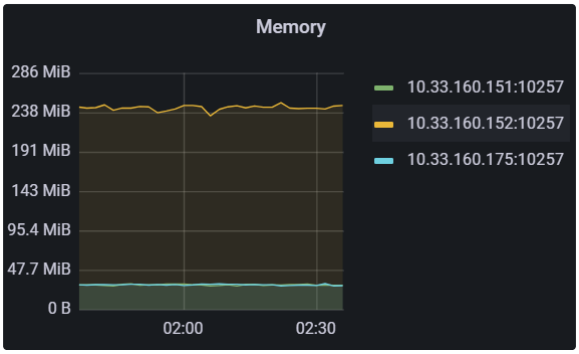
Post Request Latency 99th Quantile: This panel shows the latency experienced by the 99th percentile of POST requests made by the Controller Manager to the Kubernetes API server. This panel provides insights into the worst-case latency scenarios, highlighting the time taken for the slowest 1% of POST requests to complete. This focus on the 99th percentile helps in maintaining high performance and reliability, especially during peak loads.



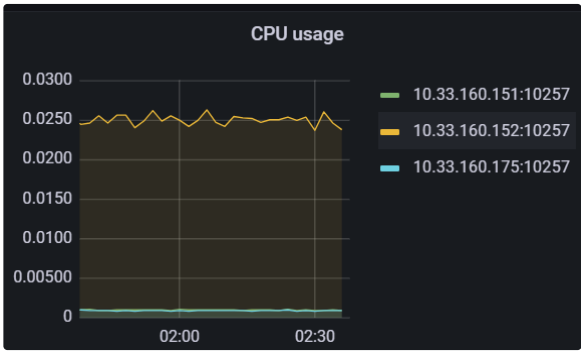
Get Request Latency 99th Quantile: This panel provides metrics on the latency of GET requests made by the Controller Manager to the Kubernetes API server, focusing on the 99th percentile. This means it highlights the latency experienced by the slowest 1% of GET requests. Tracking the 99th percentile latency helps ensure that the system can handle peak loads effectively, maintaining high performance and reliability within the Kubernetes control plane.



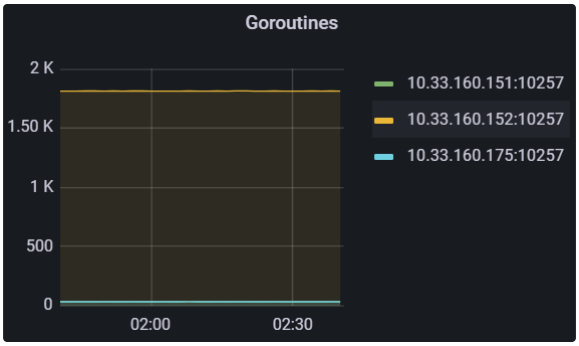
Memory: This panel provides insights into the memory usage and performance of the Controller Manager component. It typically displays metrics such as total memory consumption, memory utilization trends over time, and available memory.



CPU Usage: This panel provides insights into the CPU utilization and performance of the Controller Manager component over time. It typically includes metrics such as CPU usage trends, CPU load, and CPU utilization rates.



Goroutines: This panel offers insights into the concurrency and workload handling capacity of the Controller Manager. It typically displays metrics representing the number of Goroutines—lightweight threads managed by Go runtime—actively running within the Controller Manager at any given time.



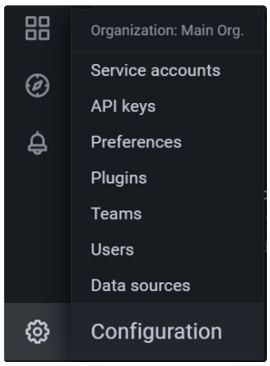
Exploring the Grafana Dashboard

To explore logs, metrics, and traces in the Grafana dashboard, click https://grafana.com/docs/grafana/latest/explore/.

Configuration of the Grafana Dashboard with Prometheus Database

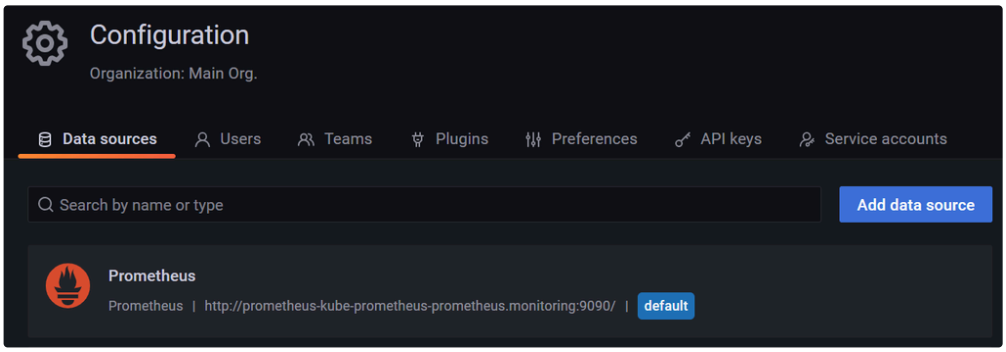
Follow these steps to configure the Grafana dashboard with Prometheus database:

1. From the Grafana sidebar menu, Go to **Configuration** > **Data sources**.

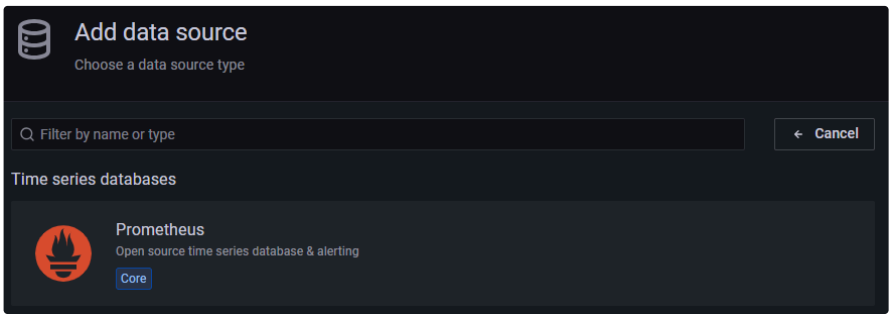


   The configuration page opens.

2. Click **Add data source.**

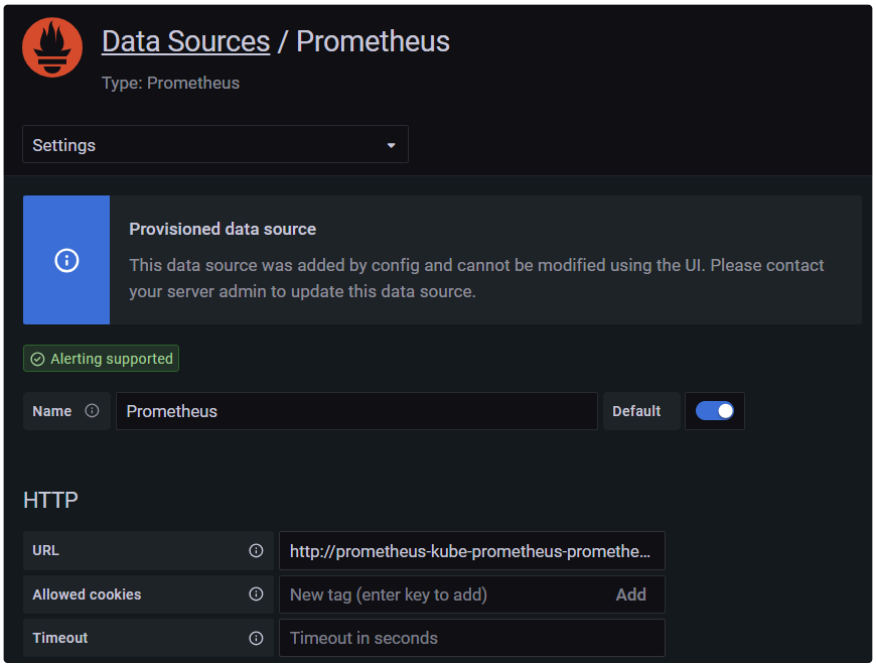


3. Click **Prometheus** as the data source type.



   The configuration page for Prometheus opens.

4. Enter the appropriate Prometheus server in the URL field (for example, http://prometheus-kube-prometheus-prometheus.monitoring:9090/).



5. Click **Save & Test** icon.
   Grafana tests the connection with Prometheus and finishes adding the data source.

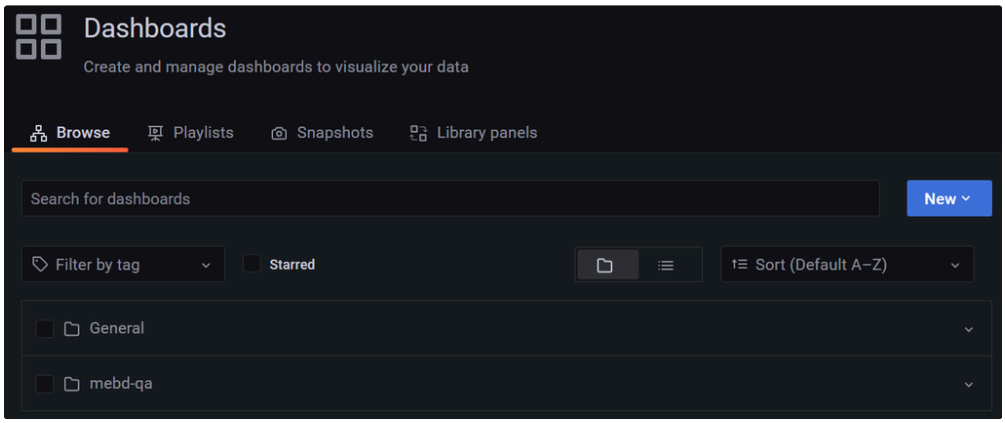
Checking the Grafana Dashboard

Follow these steps to check the Grafana dashboard:

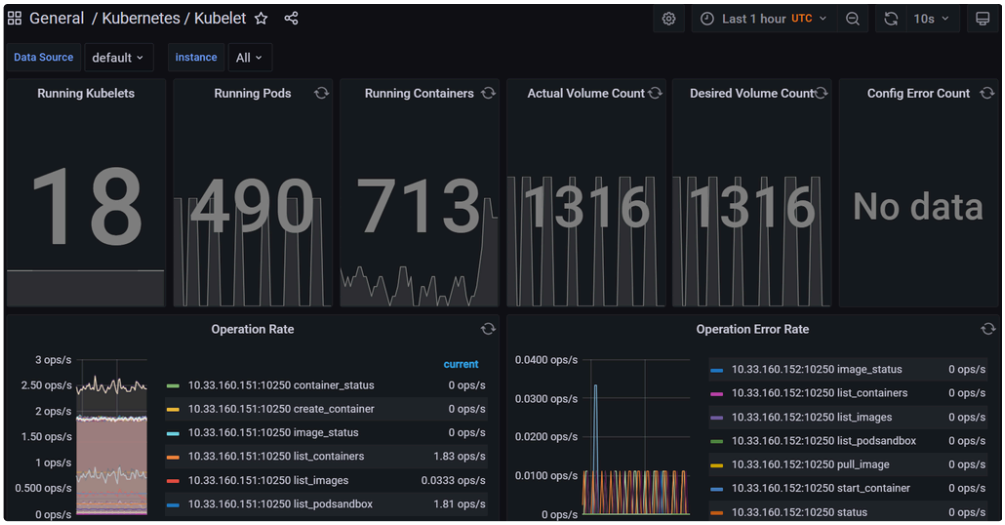
1. From the Grafana sidebar menu, Go to **Dashboards** > **Browse**.

> **Note**: You can view the dashboard by clicking the dashboard name if it displays on the home screen.

2. Enter the dashboard name to search for the dashboard.



3. Select the dashboard and review the metrics and visualizations displayed on it.
   These may include graphs, gauges, tables, or other visual elements depending on how the dashboard was configured.



4. Interact with the Dashboard depending on the configuration.
   This could involve adjusting time ranges, applying filters, zooming in on specific data points, or drilling down into more detailed views.

> **Note**: If the dashboard is configured with alerting rules, check triggered alerts. These may be displayed directly on the dashboard or accessed through the alerting menu in the Grafana.

5. Verify the data sources connected to the dashboard are functioning correctly.
   If Prometheus or any other data source is used, check that it is accessible and providing the expected metrics.
6. Inspect the settings of individual panels on the dashboard. Verify that the metrics, queries, and visualization options are configured correctly.
7. Test the functionality of the following interactive elements:
   - Buttons
   - Dropdowns or Drilldown links
8. If the dashboard is updated regularly with new features or metrics, review any recent changes to ensure they meet the requirements and expectations.

> **Note**: If you notice any issues or areas for improvement, provide feedback to the dashboard creator or make adjustments to the dashboard settings as necessary.

Alert Configuration

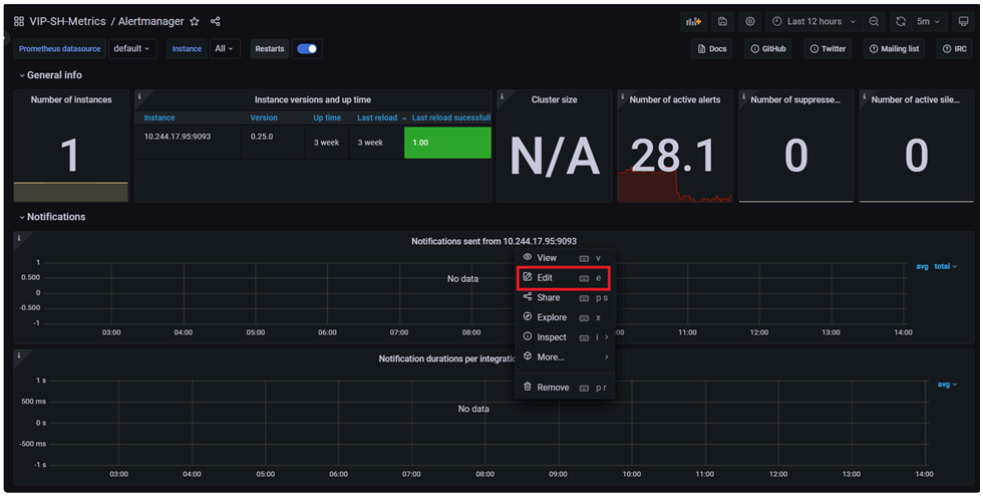
Alert configuration in Grafana involves setting up rules to monitor metrics and trigger notifications based on specific conditions.

Follow these steps to configure alerts in a Grafana dashboard:

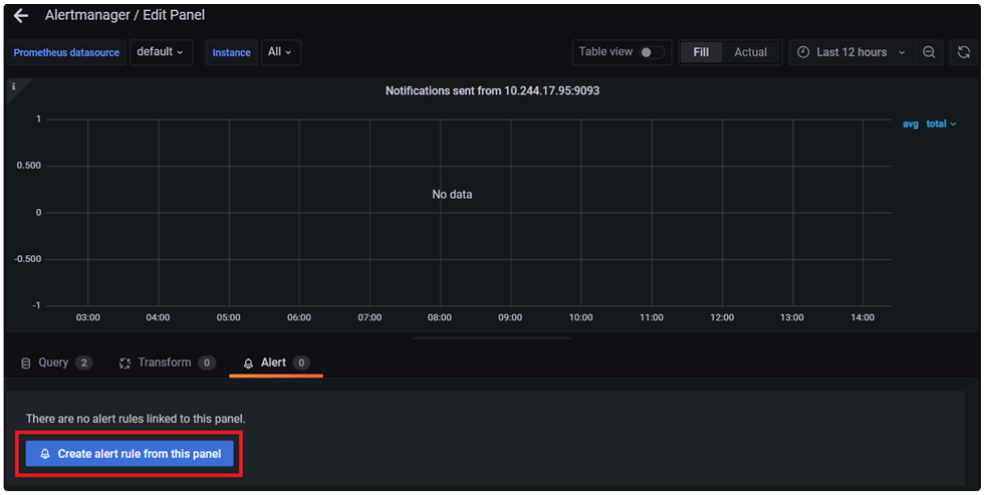
> **Note**: Before setting up alerts, ensure that your data source supports alerting. Prometheus, InfluxDB, Graphite, and Elasticsearch are commonly used data sources that support alerting in Grafana.

1. In Grafana, navigate to the dashboard that contains the metric you want to set an alert.

2. Click on the panel containing the metric and select **Edit** from the dropdown menu.



3. In the "Edit Panel" view, click on the "Alert" tab, then click **Create alert rule from this panel** icon to add a new alert rule. This will open the "Alert rules" page.

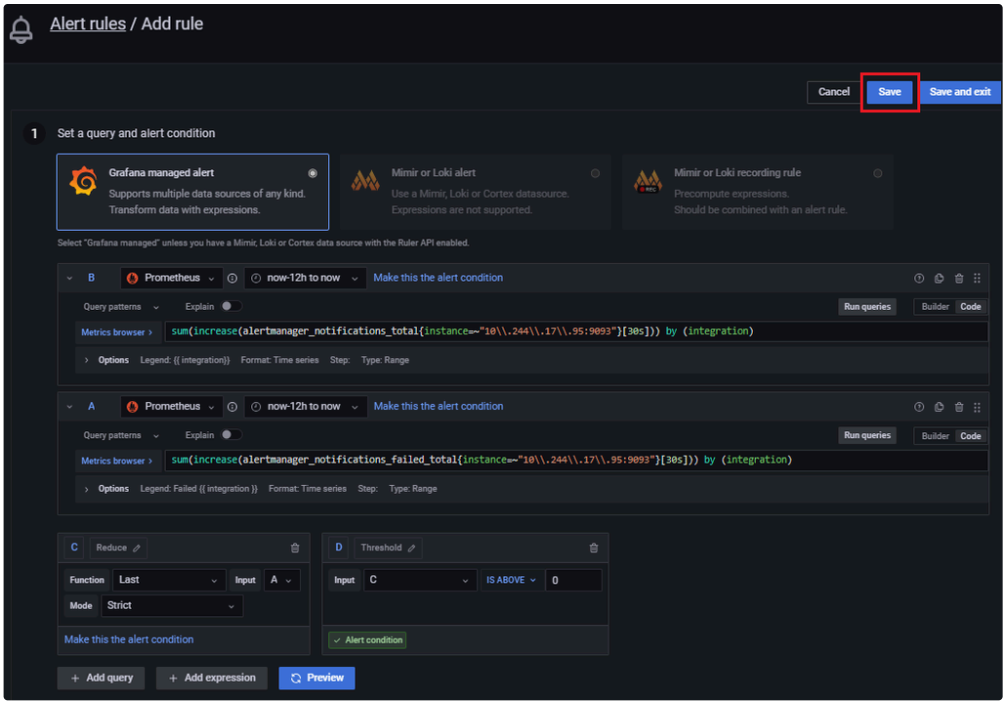


4. In the alert rules page, configure the conditions that will trigger the alert.
   This includes selecting the data source, specifying the metric or query to monitor, setting the evaluation frequency, and defining the conditions that indicate a problem (e.g., threshold crossing, absence of data).
5. Define the thresholds or conditions that will trigger the alert.
   This could be a static threshold (e.g., CPU usage exceeds 90%) or a dynamic threshold based on statistical functions (e.g., average CPU usage exceeds two standard deviations from the mean).
6. Choose how you want to be notified when the alert is triggered.
   The Grafana supports various notification channels, including email, Slack, PagerDuty, and webhook integrations. Configure the notification settings, such as recipients, message format, and severity level.

> **Note**: To ensure the alert rule is working as expected, test the alert rule by simulating an alert trigger based on historical data or current conditions to verify that the alerting mechanism is functioning correctly.

7. Click **Save** to create the alert rule.

   Give the alert rule a descriptive name and optionally add any additional labels or annotations to help identify it.



8. Toggle the "Alerting" switch in the dashboard settings to enable alerts for the dashboard.
9. Monitor the dashboard for any triggered alerts.
   If an alert is triggered, Grafana will notify the specified recipients via the configured notification channels.
10. Review and adjust the alerting settings periodically as needed.
    This may involve refining alert thresholds, adding new alert rules, or updating notification channels based on changing monitoring requirements.

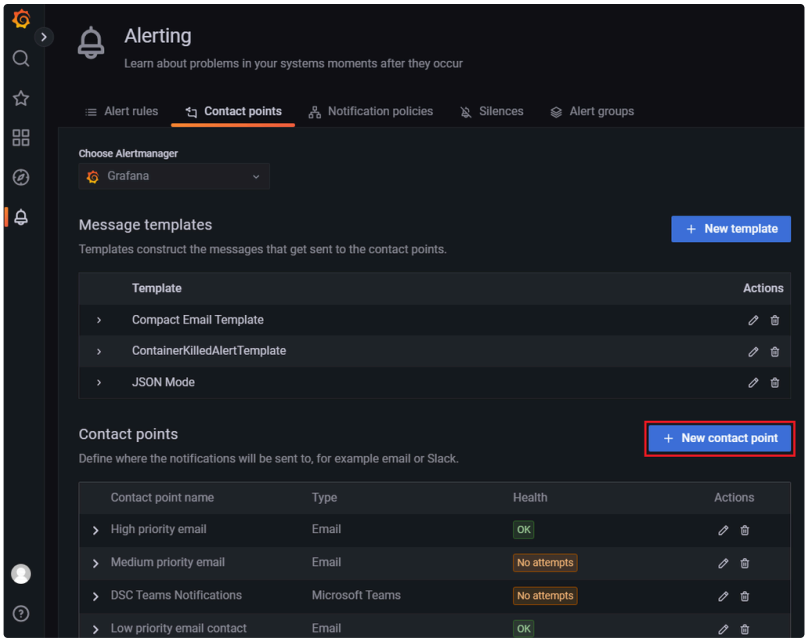
Channel Configuration to get alerts through Email

The Grafana alerting email integration sends notifications via email when your alerts are triggered. An email notification is sent when an alert is activated and when it is resolved.

> **Note**: To configure Grafana to send alerts through email, you need to set up an Email Notification Channel.

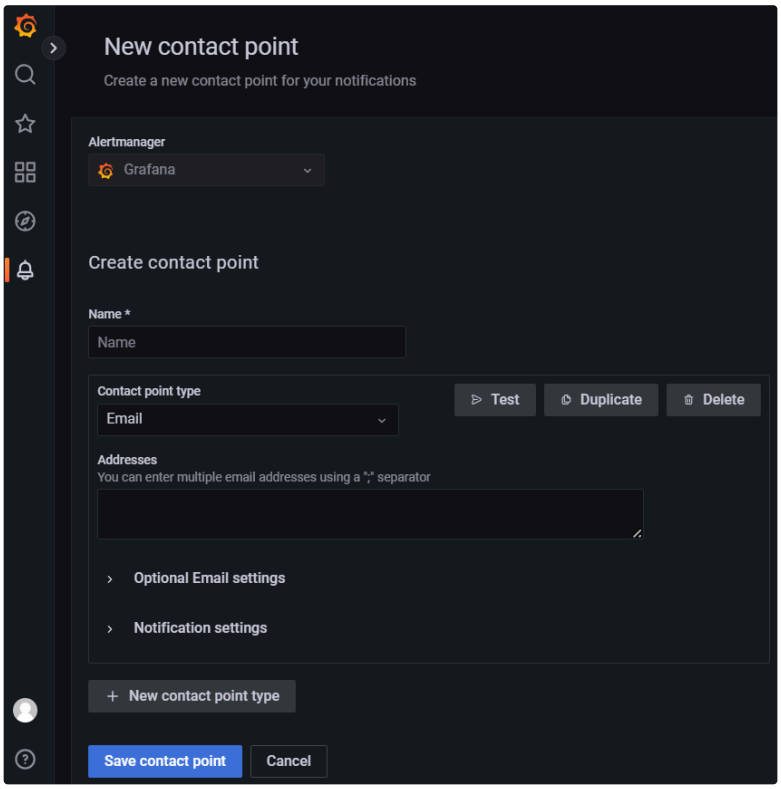
Following these steps to create an email notification channel:

1. In Grafana, navigate to **Alerting → Contact points**.
2. Click **+ New contact point**.



3. Enter the following contact point details:
   - **Name**: Enter a name for the channel (e.g., "Email Alerts").
   - **Contact point type**: Select **Email** from the dropdown list.
   - **Addresses**: Enter the email addresses to receive the alerts, separated by commas.

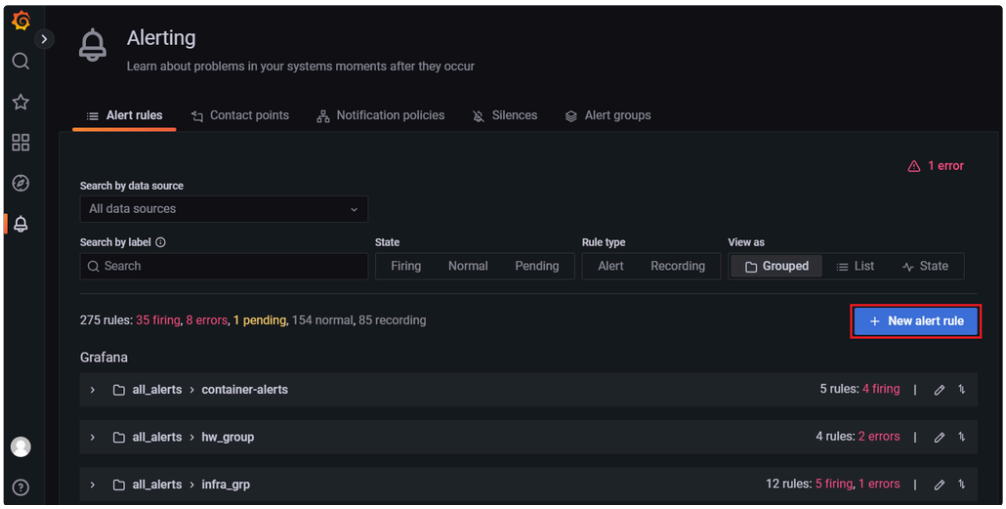
Note: E-mail addresses are case sensitive. Ensure that the e-mail address entered is correct.



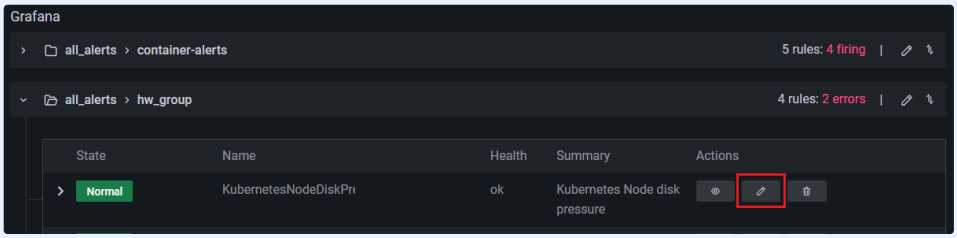
4. Click **Test** to send a test email.
   This is applicable for Grafana Alertmanager only.
5. Click **Save contact point** to create the notification channel.
   The email contact point is ready to receive alert notifications.

Follow these steps to assign the notification channel to your alert:

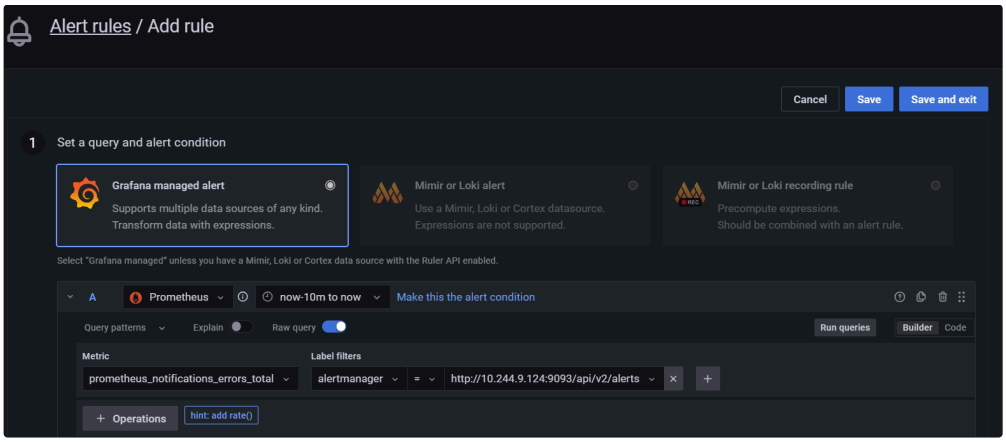
1. In Grafana, navigate to **Alerting** → **Alert rules**.
2. Click **+ New alert rule**.



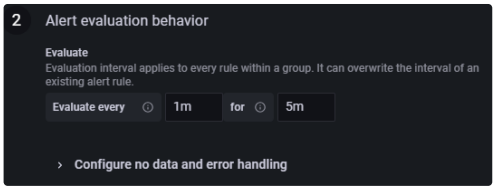
- The Grafana managed alert option is selected by default.
- To modify the details of an existing alert, select the alert and click the **Edit** icon.



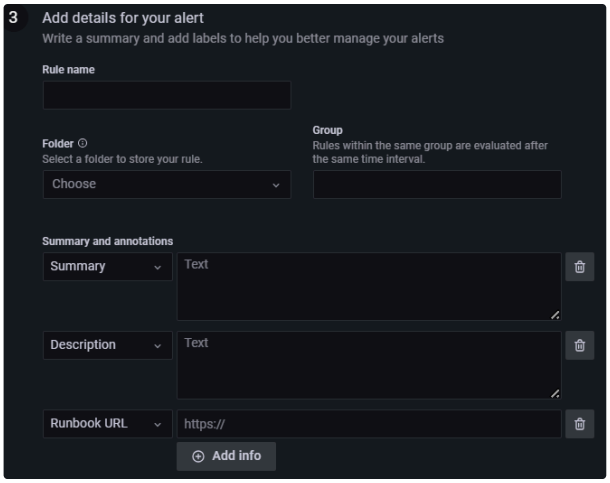
3. Define your query and set the alert condition.



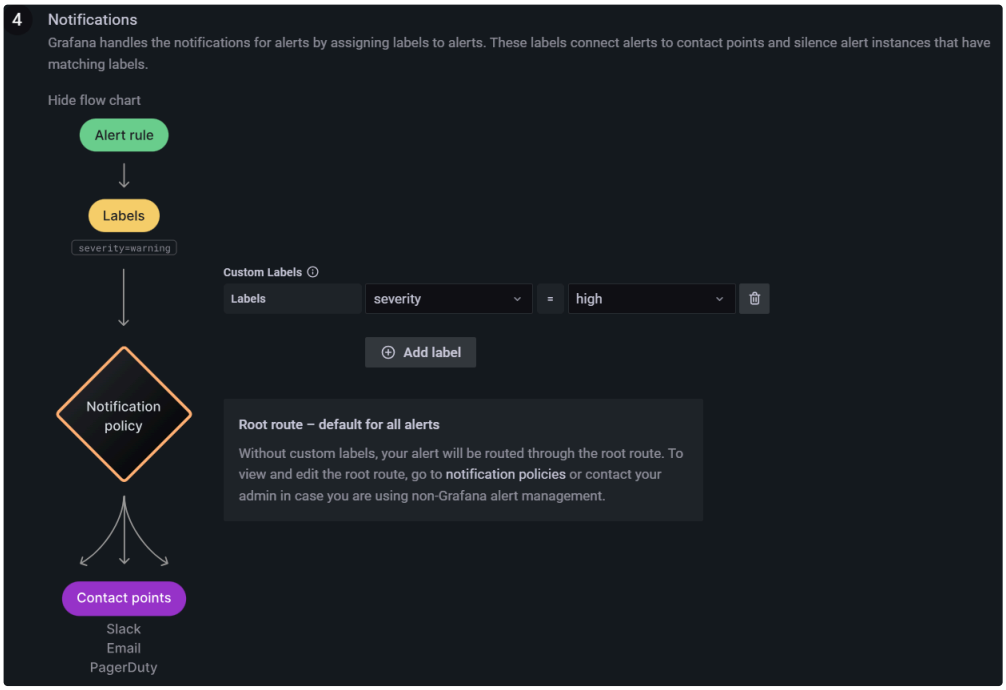
4. Set the evaluation interval for the alert rule.



5. Provide a rule name and select a folder to store your rule.



6. Add a summary, labels, and annotations to manage your alerts better.
7. Go to the **Notifications** section.



8. Add custom labels to connect alerts to contact points.

> **Note**: Without custom labels, alerts will be routed through the root route.

9. Click **Save and exit** to finalize your alert rule.

> If emails are not being sent, check Grafana logs for errors.

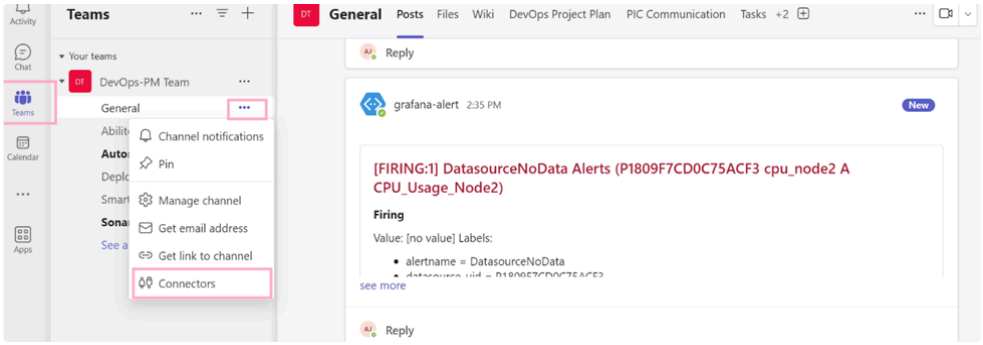
Channel Configuration to get alerts through Microsoft Teams

> **Note**: Ensure that you have set up alerting rules in Grafana for the metrics you want to monitor.

> **Note**: The webhook integration must be set between Grafana and Microsoft Teams to receive alerts from Grafana in Microsoft Teams.

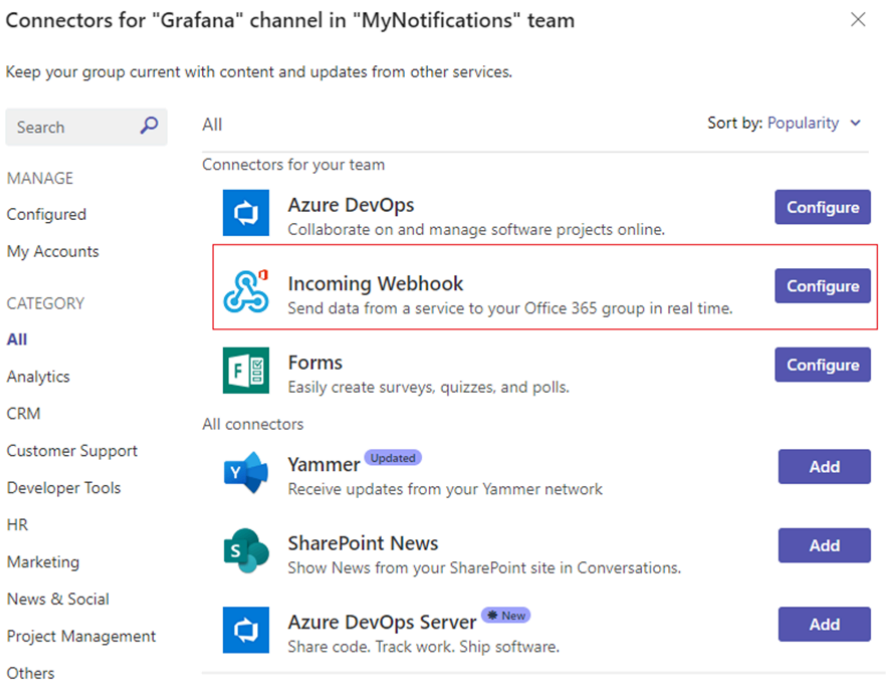
Follow these steps to create an Incoming Webhook in Microsoft Teams:

1. Open Microsoft Teams and navigate to the channel where you want to receive alerts.
2. Click on the three dots or ellipsis (**...**) next to the channel name and select **Connectors**.

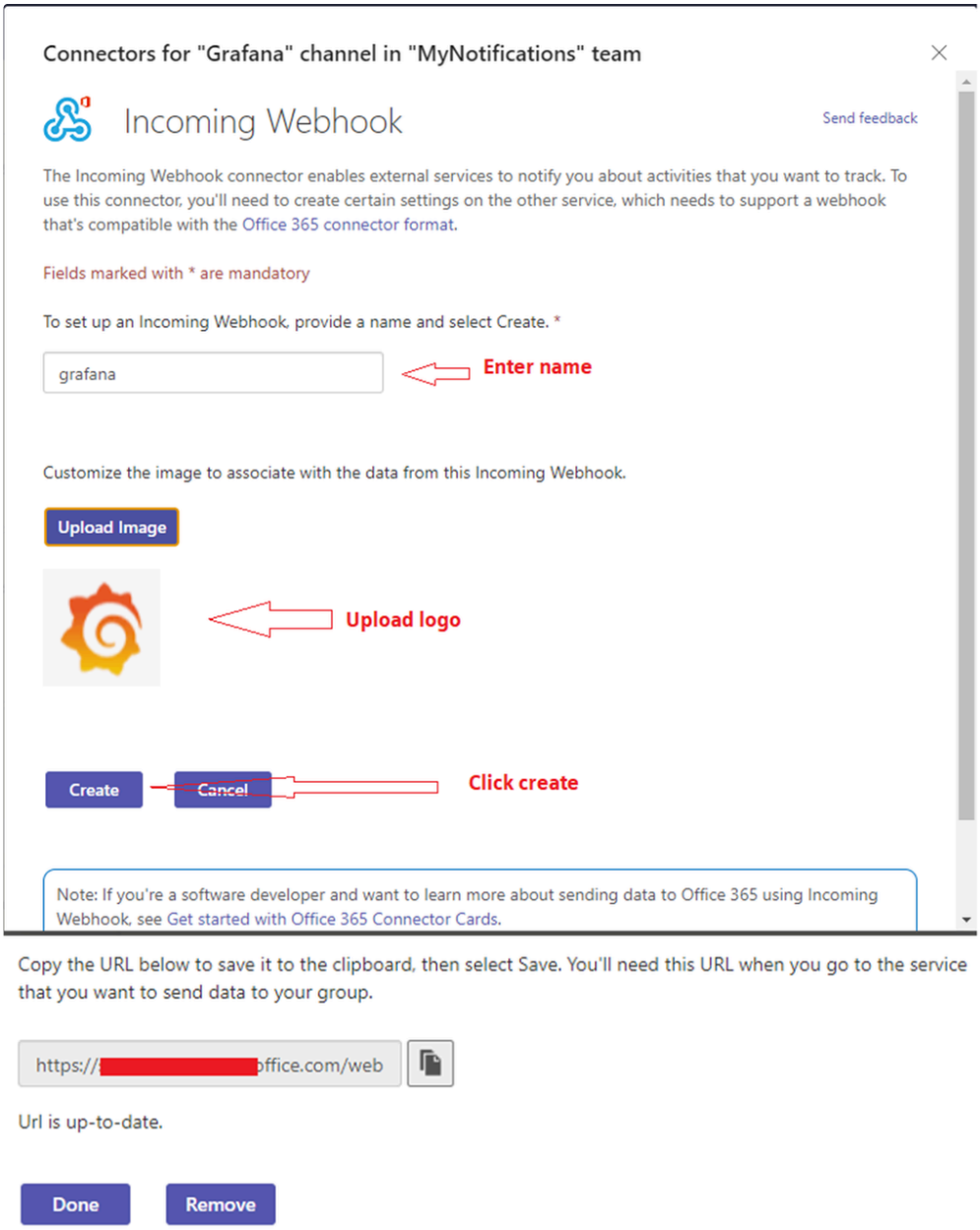


3. Search for "Incoming Webhook" and click on it to add the connector.

> **Note**: The webhook integration must be set between Grafana and Microsoft Teams to receive alerts from Grafana in Microsoft Teams.



4. Follow the prompts and copy the generated webhook URL.



Follow these steps to configure Alert Notifications in Grafana:

1. In Grafana, navigate to the dashboard where you have set up alerts.
2. Click on the gear icon in the upper-right corner of the dashboard to access the dashboard settings, then select the "Notification channels" tab.
3. Click on "Add channel" and select "Microsoft Teams" from the list of available notification channels.
4. Paste the webhook URL you copied from Teams into the "URL" field.
5. Click "Save" to add the Teams notification channel.

Follow these steps to test the alert:

Note: Testing the notification channel ensures that alerts from Grafana are sent to Teams.

1. Trigger an alert on your Grafana dashboard (e.g., by temporarily modifying a threshold to exceed the current value).
2. Check the Microsoft Teams channel where you configured the webhook to verify that the alert notification is received.
3. Monitor the alert notifications in Teams to ensure that you are receiving them as expected.

4. If you encounter any issues with alert delivery or if alerts are not triggering as expected, review the configuration settings in both Grafana and Teams to troubleshoot the problem.

Channel Configuration to get alerts through Third Party Applications

> **Note**: To configure alerts from Grafana to third-party applications, such as Slack, PagerDuty, and email, you can utilize webhook integrations or specific notification channels provided by Grafana.

Follow these steps to set up an integration in third party applications:

1. Choose the third-party application you want to receive alerts.
   The common options include Slack, PagerDuty, Microsoft Teams, and email.
2. Set up an integration or webhook by following the documentation provided by the third-party application. This usually involves generating a unique URL or API key that Grafana will use to send alerts to the application.

Follow these steps to configure the notification channel in Grafana:

1. In Grafana, navigate to the dashboard where you have set up alerts.
2. Click on the gear icon in the upper-right corner of the dashboard to access the dashboard settings, then select the "Notification channels" tab.
3. Click on "Add channel" and select the appropriate notification channel for the third-party application (e.g., Slack, PagerDuty, Email).
4. Enter the necessary configuration details, such as the webhook URL or API key provided by the third-party application.
5. Click "Save" to add the notification channel.

Follow these steps to test the alert:

> **Note**: Testing the notification channel ensures that alerts from Grafana are sent to the third-party application.

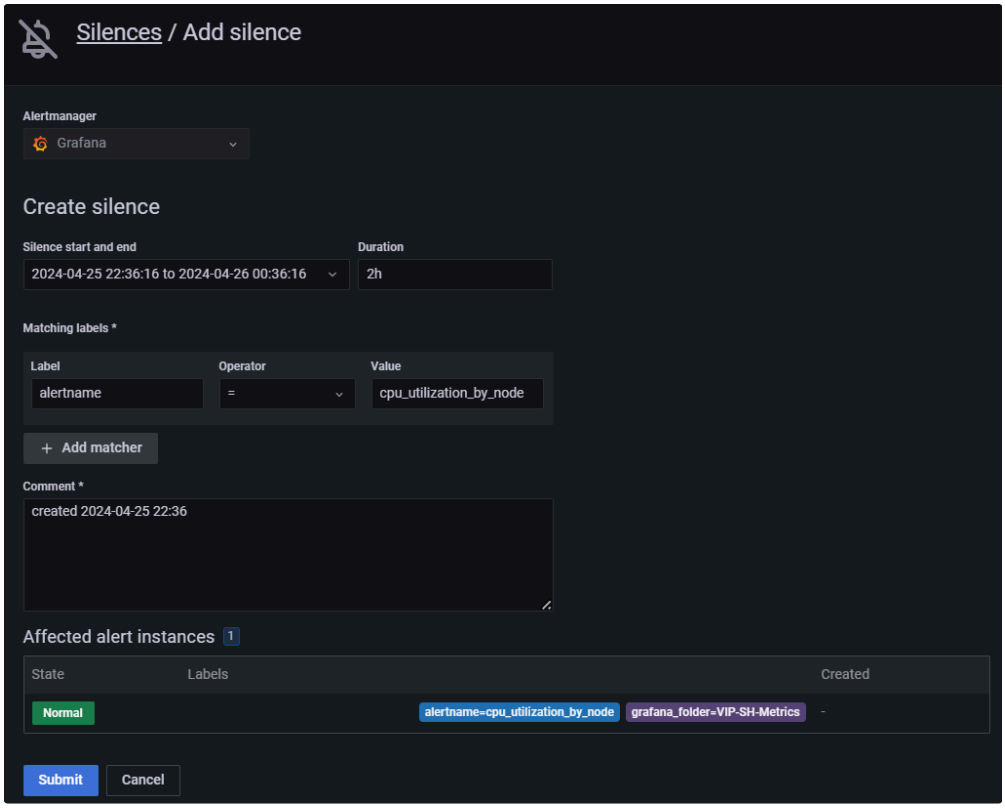
1. Trigger an alert on your Grafana dashboard (e.g., by temporarily modifying a threshold to exceed the current value).
2. Check the third-party application (e.g., Slack channel, PagerDuty incident, email inbox) to verify that the alert notification is received.
3. Monitor the alert notifications in the third-party application to ensure that you are receiving them as expected.
4. If you encounter any issues with alert delivery or if alerts are not triggering as expected, review the configuration settings in both Grafana and the third-party application to troubleshoot the problem.

Pausing an Alert

An alert can be paused temporarily in Grafana by modifying the alerting rules or configurations in the monitoring system.

Follow these steps to pause an alert:

1. In Grafana, Access the alerting rules.
2. Identify the alert rule corresponding to the alert you want to pause.
3. Comment out the entire rule by adding a '#' at the beginning of the line, effectively disabling it.
4. Save the changes to the Prometheus configuration file.



5. Reload the Prometheus configuration to apply the updates.
   This process may vary depending on your Prometheus setup, but typically involves sending a SIGHUP signal to the Prometheus process.
6. Check the monitoring system to ensure that the alert is no longer firing.
7. Confirm that the alert rule has been commented out and the alert is effectively paused.

> **Note**: If the alert needs to be resumed, reverse the changes made to the alerting rule by uncommenting the rule and reloading the Prometheus configuration.

Modifying an Alert

An alert can be modified in Grafana by adjusting the alerting rules or configurations in your monitoring system, such as Prometheus.

Follow these steps to modify an alert:

1. Access the alerting rules defined in the Prometheus configuration file (*prometheus.yml*).
2. Identify the alert rule corresponding to the alert you want to modify.
3. Modify the alert rule to adjust the conditions, thresholds, labels, annotations, or other parameters as needed. You can change the Prometheus expression (*expr*), duration (*for*), severity labels (*labels*), annotations (*annotations*), and other properties of the alert rule.
   For example:
   - *alert: HighCPUUsage*
     *expr: node_cpu_seconds_total / ignoring(cpu) node_cpu_seconds_total{mode="idle"} * 100 > 90*
     for: 10m
     labels:
       severity: critical
     annotations:
       summary: "High CPU usage detected"
       description: "The CPU usage on the server is above 90% for more than 10 minutes."

4. Save the changes to the Prometheus configuration file.
5. Reload the Prometheus configuration to apply the updates.
   This process may involve sending a SIGHUP signal to the Prometheus process or using the Prometheus API to reload configuration dynamically.
6. Check the monitoring system to ensure that the modified alert behaves as expected.
7. Confirm that the changes made to the alert rule are reflected in the alerting system.

> **Note**: Trigger the alert to verify that it fires according to the modified conditions. It can be done by simulating conditions that meet the alert criteria or by temporarily modifying the alerting rule to trigger the alert.

8. Review the performance and behavior of the modified alert over time. Adjust the alerting rules or configurations further if necessary to optimize alerting accuracy and effectiveness.

Application's Health Monitoring through Grafana Dashboard

Monitoring an application's health through a Grafana dashboard involves collecting, visualizing, and analyzing various metrics related to the application's performance, deployment availability, pod’s health status, cron job status, service availability, cluster health, worker node health, and control plane health. By setting effective health monitoring, you can proactively identify issues, troubleshoot problems, and ensure the reliability and performance of the application.

Follow these steps to monitor an application’s health through a Grafana dashboard:

1. Determine which metrics are critical for assessing the application's health. These may include:
   - Response time: Average, median, and maximum response times for HTTP requests.
   - Error rates: Percentage of requests that result in errors (e.g., 4xx, 5xx status codes).
   - Throughput: Number of requests processed per second or minute.
   - Resource usage: Utilization of CPU, memory, disk, and network.
   - Service dependencies: Health status of external services or dependencies the application relies on.
2. Choose metrics that reflect the performance, availability, and reliability of the application.
3. Connect Grafana to your data sources where these metrics are collected.
   The common data sources include Prometheus, InfluxDB, Graphite, Elasticsearch, and others.
4. Configure data source settings in Grafana to establish the connection and retrieve metrics.
5. Create a new dedicated dashboard in Grafana to monitor the application's health.
6. Decide on the layout and organization of the dashboard.
   Consider grouping related metrics together and arranging them logically for easy monitoring.
7. Add panels to the dashboard to visualize the metrics data.
   Choose appropriate visualization types (e.g., graphs, gauges, tables) based on the nature of the metric and the monitoring objectives.
8. Configure alerts as per your requirements to get notified.
9. Specify notification channels (e.g., email, Slack, PagerDuty) where alert notifications should be sent.
10. Test the dashboard and alerting rules to ensure they are working as expected.
    Manually trigger alerts if necessary to verify that notifications are sent correctly.
11. Monitor the dashboard in real-time to confirm that metrics are displayed accurately, and that the dashboard provides the required insights.
12. Periodically review and update the dashboard and alerting configuration as the application evolves.
13. Adjust alerting thresholds or add new metrics to capture changes in application behavior or performance requirements.

Note: To achieve the above-mentioned critical metrics, you need to write the PromQL on the dashboards. For example: `sum(alertmanager_alerts{namespace=~"monitoring",service=~"prometheus-kube-prometheus-alertmanager"}) by (namespace,service,instance)`

Infrastructure Monitoring

Infrastructure monitoring involves tracking and visualizing various metrics related to the IT infrastructure, including servers, networks, databases, and other components. There is a namespace filter on this dashboard from the user can select the desired namespace for monitoring. This namespace will include Infrastructure as well as application namespaces. Based on selection, the user can see the desired metrics.

Follow these steps to check the infrastructure monitoring:

1. In Grafana, navigate to the dashboard that contains the infrastructure monitoring panels.
   This dashboard may be a pre-configured dashboard provided by Grafana or a custom dashboard you have created.
2. Check the panels on the dashboard to review key metrics related to your infrastructure. These metrics may include CPU usage, memory usage, disk space, network traffic, system health, and more.

Note: Pay attention to any graphs, gauges, or tables that visualize these metrics and provide insights into their current values and trends over time.

3. Scan the panels for any anomalies or issues that may indicate problems with your infrastructure. Look for spikes or dips in metrics that could signal performance bottlenecks, resource constraints, or system failures.

Note: Pay special attention to any panels that have triggered alerts or crossed predefined threshold values.

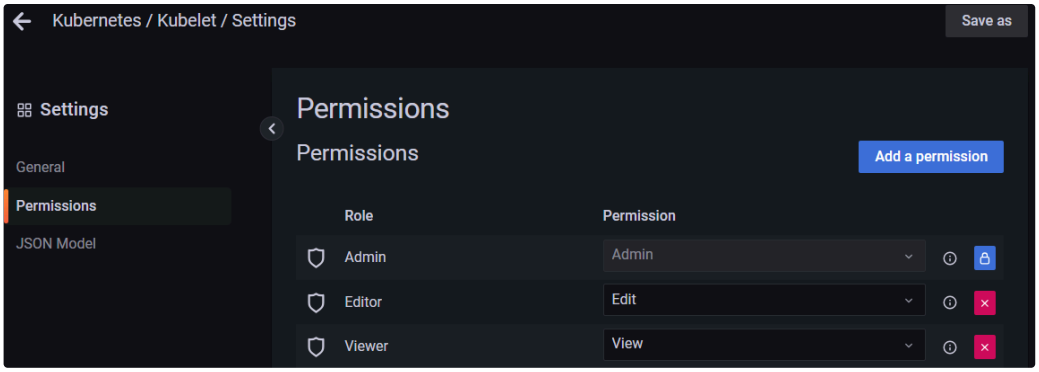
4. Review the alert conditions, thresholds, and notification channels configured for each alert rule to determine the appropriate response.
5. Hover over data points on graphs or click on panels to view additional details and context about the metrics being monitored.
6. Monitor key metrics over time to identify trends, patterns, or recurring issues that may require attention.
7. Continuously refine and adjust the monitoring setup, dashboard layouts, alerting rules, and thresholds to improve the effectiveness of the infrastructure monitoring efforts.

Dashboard Permissions

When you want to extend a viewer’s ability to edit and save dashboard changes or limit an editor’s permission to modify a dashboard, you can assign permissions to dashboards and folders. For example, you might want a certain viewer to be able to edit a dashboard. While that user can see all dashboards, you can grant them access to update only one of them.

Follow these steps to access the dashboard user permissions:

1. In Grafana, navigate to the dashboard where you want to set the dashboard permissions.
2. Click on the Dashboard settings (⚙️) icon on the upper-right corner of the dashboard to access the dashboard settings, then select the "Permissions" tab.



3. You can specify the following permissions to dashboards and folders from the dropdown list:

- **Admin**: Can create, edit, or delete a dashboard. Can edit or delete a folder and create dashboards and subfolders in a folder. Administrators can change dashboard and folder permissions.
- **Edit**: Can create, edit, or delete a dashboard. Can edit or delete a folder and create dashboards and subfolders in a folder. Editors cannot change folder or dashboard permissions.
- **View**: Can only view dashboards and folders.

Note: When a user creates a dashboard or a folder, they are automatically granted **Admin** permissions for it.

Appendix

IF Adapter

List of IF Adapters

The following are the list of IF adapters:

Adapter Name	Description
panasonic-generic-csv-cifs-watch-adapter	It is a generic adapter that reads CSV files.
Panasonic.PPE.Inspection.Result.Adapter	It is a manual QC adapter.
Panasonic.MES.Rewinding.Adapter	It is an unstable slitting and normal rewind adapter.
Panasonic.MES.Api.Adapter	It is a coating, pressing, and sunburn adapter.
Panasonic.MES.Slitting.Adapter	It is a Slitting/Pancake adapter (Normal + IL).
panasonic.mes.winding.adapter	Not in use
panasonic.mes.assembly.adapter	Not in use
panasonic.formation.file.adapter	It is a formation adapter-different contracts are handled.
panasonic.mes.arp.adapter	It is an Address Resolution Protocol (ARP) adapter-FA101.
panasonic.mes.arpdrop.adapter	It is an ARP Drop adapter-FA102.
panasonic-vi-file-adapter	Not in use
panasonic-mes-defectrollup-adapter	It is a defect rollup adapter.
panasonic-mes-csv-file-adapter	Not in use
panasonic-mes-mixing-adapter	Mixing is first step, and this mixture is called slurry and is made up of active materials, conductive additives, solvents, and binders.
panasonic-assembly-file-adapter	It is an assembly adapter-different contracts are handled.
panasonic-mes-timer-watch-adapter	It is a timer expiration adapter.
panasonic-pitch-file-adapter	Not in use
panasonic-mes-api-process-adapter	It is a process completion adapter.
panasonic-packaging-adapter	It is a packaging adapter-different contracts are handled.
panasonic-ifl-file-clean-adapter	It is a clean backup file SMB adapter.
panasonic-cyclecount-erp-adapter	It is an ERP cycle count adapter.
panasonic-erp-fileread-adapter	It is an ERP file read adapter.

panasonic-erp-filewrite-adapter	It is an ERP file write adapter.
panasonic-wcs-integration-adapter	It is a WCS API integration adapter.
panasonic-kafkaclient-nodered-t1	It is a Kafka consumer adapter.
panasonic-generic-edgepc-api-adapter-t1	It is an Edge PC generic API adapter.
panasonic.mes.winding.assembly.adapter	It is a winding, assembly can adapter.
panasonic-mes-optimize-pitch-adapter	It is an optimized pitch adapter-different contracts are handled.
panasonic-mes-evaluation-code-adapter	It is an evaluation code adapter.
panasonic.qc.inspection.adapter	It is a daily and monthly QC inspection adapter.

Connection Configuration Files

The following are the links to download the connection configuration file:

Setup Integrations	Document	Link	PIC
ERP	ERP Data Contract	https://pcus1.sharepoint.com/:x:/s/MEBDMESProjectwithHCL/ERGT_KA3LxVJg7uBEdGdbVABAzPiFisb7qqygSw3IpaMIQ?e=Jq5GKf	tamizhselvan.seethaaman@ext.us.panasonic.com
	ERP Bitbucket repository	https://bitbucket.org/panasonicdsc/panasonic.integration.adapter.configurations/src/master/MES-ERP/	
	Adapter connection configuration	MES-ERP-Iritode_Adaptor_Configurations.xlsx	
PPEGW		MES_Adapter_Configuration-latest_version.xlsx	pakhi.jain@ext.us.panasonic.com
		Master Data Setup_(Dev) - V2 -Copy.xlsx	
	WCS Data Contract	https://pcus1.sharepoint.com/:x:/s/MEBDSolutionProject/ESsjRfOkTqJMv7SwaB6UaMEB0HoBnuYwibdA1TDoy5-SlA?e=Lin1B1	

WCS	PCNA Test Case Documents	PCNA Test Case Documents	sai.hemanth@ext.us.panasonic.com
	WCS Attributes	WCS Attributes.xlsx	
	WCS API Integration Configurations	WCS-API-Integration-Configurations.xlsx	
	WCS File Adapter Configurations	WCS-File-Adapter-Configurations.xlsx	
EDGE		MES_Adapter_Configuration-latest_version.xlsx	pakhi.jain@ext.us.panasonic.com
		Master Data Setup (Dev) - V2 - Copy.xlsx	
F-SYS		MES_Adapter_Configuration-latest_version.xlsx	
		Master Data Setup (Dev) - V2 - Copy.xlsx	
FACILITES (OPCUA)			
IRITODE	Adapter connection configuration	MES-ERP-Iritode Adaptor Configurations.xlsx	tamizhselvan.seetharaman@ext.us.panasonic.com
SCADA			
DATA SPIDER COA SERVER			
QC (Offline)		MES_Adapter_Configuration-latest_version.xlsx	
QC (Daily/Monthly)	Daily MonthlyQC (connection configuration)	DailyMonthlyQC.xlsx	
	CloudQADruid.postman_c ollection.json		
	delta-V Soda Report -		

SODA DeltaV	Copy.docx Mebd_Druid_postman_collection_with_env.json SODADeltaQuery.sql Example of Data Get from SmartHive.pptx	SODA DeltaV	vikash.vitthore@ext.us.panasonic.com
Packing SYS			
Label Printing			

Description of the Adapter Fields

Field Name	Description	Sample Field Values
Name	This refers to the name of the connection.	*“masterdata-erp-adaptor”*
Integration Type	This refers to the type of integration.	*“All”*
Modified	This refers to the number of days when the adapter configuration was modified. **Note:** This field is available only when editing the adapter connection configuration.	
Adapter Type	This refers to the type of adapter. **Note:** Each adapter type has its own configuration and processing flows.	*“Smart Storage Integration, Smart Storage File, or MMS Integration”*
Adapter	This refers to the name of the adapter used to create this connection.	*“panasonic-erp-integration-adapter”*
Adapter Version	This refers to the latest version of the adapter.	*“0.0.5”*
Description	This refers to the basic description of the adapter. It helps other users understand the purpose of the adaptor and can help simplify adaptor search.	*“This adapter is a medium to set up communication between the ERP and the MES (master data) “.*

<table>
<tr><td rowspan="2">Size</td><td>This refers to the size of the adapter. The size (resource requested) refers to a pre-configured amount of CPU and memory allocated to the database. This allocation will be used as a parameter for requests and limits during deployment.</td><td>“XS, S, M, and L.”</td></tr>
<tr><td colspan="2">Below is a description of what each size represents, including the CPU/memory request and limit for each one:
<table><tr><th>Size</th><th>Description</th><th>CPU Request</th><th>CPU Limit</th><th>Memory Request</th><th>Memory Limit</th></tr><tr><td>XS</td><td>Extra Small</td><td>50m</td><td>75m</td><td>100Mi</td><td>150Mi</td></tr><tr><td>S</td><td>Small</td><td>100m</td><td>150m</td><td>150Mi</td><td>225Mi</td></tr><tr><td>M</td><td>Medium</td><td>150m</td><td>225m</td><td>300Mi</td><td>450Mi</td></tr><tr><td>L</td><td>Large</td><td>300m</td><td>450m</td><td>600Mi</td><td>900Mi</td></tr></table>The unit suffix M represents milli cores (m), and Mi denotes mebibytes (Mi).</td></tr>
<tr><td>Status</td><td>It displays the status of the connection. Use the toggle to enable or disable the adapter connection.</td><td>Enabled toggle: Enables the adapter to start communication between ERP and the MES applications.
Disabled toggle: Stops the adapter communication between ERP and the MES applications.</td></tr>
<tr><td>Adapter Name</td><td>This refers to the name of the adapter.</td><td>“Panasonic-ERP-Integration-Adapter”</td></tr>
<tr><td>Elastic Search URL</td><td>This refers to the URL location of the Elastic search where the data is stored and indexed.</td><td>“http://elastic:kW1T74pcH5V715vP5Ad42Afs@sm-main-elasticsearch-es-http.sm-elasticsearch.svc.cluster.local:9200”</td></tr>
<tr><td>Server IP Address or Hostname</td><td>This refers to the IP address of the server on the network.</td><td>“10.140.79.140”</td></tr>
<tr><td>Server Port</td><td>This refers to the port number where the server is listening to the request.</td><td>“22”</td></tr>
<tr><td>Server Username</td><td>This refers to the username used for authentication, logging into, and accessing the server to perform various actions.</td><td>“pssna”</td></tr>
<tr><td>Server Password</td><td>This refers to the password used along with the username to authenticate and access the server to perform various actions.</td><td>“Panasonic”</td></tr>
<tr><td>File Extension Filter</td><td>This refers to the valid file extension format to be used for the processing of the la01, la02, and la03 files.</td><td>“["dat"]”</td></tr>
<tr><td>Ignore File Extensions</td><td>This refers to the file extension format to be</td><td>“["DAT_","dat_"]”</td></tr>
</table>

	ignored for the processing of the la01, la02, and la03 files.	
Source Folder Path	This refers to the FTP folder location on the server where the la01 (la02 and la03) files are saved.	*“/home/pssna/MEBD/la01,* */home/pssna/MEBD/la02,* */home/pssna/MEBD/la03”*
Kafka Connection	This refers to the Kafka URL to establish a connection. This URL comprises of hostname and port.	“mwsinv-eh-0-dev.servicebus.windows.net:9093”
Topic Name	This refers to the name of the Topic under the Kafka Cluster to which the messages are published by producers.	*“panasonic.masterdata.inbound.topic”*
Kafka Username	This refers to the authenticated username used by the Kafka producers to securely connect to the Kafka topic and publish data on it.	*kafka*
Kafka Password	This refers to the authenticated password used by the Kafka producers to securely connect to the Kafka topic and publish data on it.	*kafka*
Kafka Authentication Type	This refers to the authorization key used by the Kafka producers to build a secure Kafka connection to publish data on the Kafka Topic.	*plaintext*
Kafka Notification CG	This refers to the name of the consumer group for notifications. This group is authorized to access the Kafka topic to process the published data related to the master data. The consumer group configuration must be unique per topic.	*“cg_mwsinv_master_adp”*

Data Contract Format - IF Adapters

Color	Significance of color in the table

	These fields are environment specific. Meaning it will be configured differently for SFDEV, MEBD-QA, MEBD-PROD, etc
	These fields are again environment specific. But these fields also need to be updated whenever you update or perform version-up in the respective coreapp. For example Workflow version needs to be updated if we upgrade the version. Similarly for Product System Version.
	This fields are according to MEBD PROD environment

Adapter Information	**Adapter Connection**							
	FV101	**FV103**	**FV106**	**FV111**	**FV112**	**FV113**	**FV114**	**FV115**
Adapter & Version	**panasonic-vi-file-adapter v1.0.1+**	**panasonic-vi-file-adapter v1.0.1+**	**panasonic-mes-winding-assembly-adapter v1.0.0+**	**panasonic-mes-winding-assembly-adapter v1.0.0+**	**panasonic-mes-winding-assembly-adapter v1.0.0+**	**panasonic-mes-winding-assembly-adapter v1.0.0+**	**panasonic-mes-winding-assembly-adapter v1.0.0+**	**panasonic-mes-winding-assembly-adapter v1.0.0+**
Configuration Field								
Name	watch-vi-fv101	watch-vi-fv103	vi-mes-fv106-optimise	vi-mes-fv111-optimise	vi-mes-fv112-optimis	vi-mes-fv113-optimise	vi-mes-fv114-optimise	watch-vi-fv115
Adapter	panasonic-vi-file-adapter	panasonic-vi-file-adapter	panasonic-mes-winding-assembly-adapter	panasonic-mes-winding-assembly-adapter	panasonic-mes-winding-assembly-adapter	panasonic-mes-winding-assembly-adapter	panasonic-mes-winding-assembly-adapter	panasonic-mes-winding-assembly-adapter
Adapter Version	1.0.0	1.0.0	1.0.0	1.0.0	1.0.0	1.0.0	1.0.0	1.0.0
Adapter Name	Panasonic-VI-File-Adapter	Panasonic-VI-File-Adapter	Panasonic-VI-File-Adapter	panasonic-mes-winding-assembly-adapter	Panasonic-VI-File-Adapter	Panasonic-VI-File-Adapter	Panasonic-VI-File-Adapter	Panasonic-VI-File-Adapter

Source Folder Path	\\10.33.174.74\public\forMESC\FV101	\\10.33.174.74\public\forMESC\FV103	\\10.33.174.74\public\forMESC\FV106	\\10.33.174.74\public\forMESC\FV111	\\10.33.174.74\public\forMESC\FV112	\\10.33.174.74\public\forMESC\FV113	\\10.33.174.74\public\forMESC\FV114	\\10.33.174.74\public\forMESC\FV115
Source Folder UserName	smb04	smb04	smb04	smb04	smb04	smb04	smb04	smb04
Source Folder Password	WakaKino04	WakaKino04	WakaKino04	WakaKino04	WakaKino04	WakaKino04	WakaKino04	WakaKino04
File Extension Filter	["csv","CSV"]	["csv","CSV"]	["csv","CSV"]	["csv","CSV"]	["csv","CSV"]	["csv","CSV"]	["csv","CSV"]	["csv","CSV"]
Ignore File Extensions	["csv_","CSV_"]	["csv_","CSV_"]	["csv_","CSV_"]	["csv_","CSV_"]	["csv_","CSV_"]	["csv_","CSV_"]	["csv_","CSV_"]	["csv_","CSV_"]
Error Folder Path	\\10.33.174.74\public\forMESC\FV101\Error	\\10.33.174.74\public\forMESC\FV103\Error	\\10.33.174.74\public\forMESC\FV106\Error	\\10.33.174.74\public\forMESC\FV111\Error	\\10.33.174.74\public\forMESC\FV112\Error	\\10.33.174.74\public\forMESC\FV113\Error	\\10.33.174.74\public\forMESC\FV114\Error	\\10.33.174.74\public\forMESC\FV115\Error
Working Folder Path	\\10.33.174.74\public\forMESC\FV101\Work	\\10.33.174.74\public\forMESC\FV103\Work	\\10.33.174.74\public\forMESC\FV106\Work	\\10.33.174.74\public\forMESC\FV111\Work	\\10.33.174.74\public\forMESC\FV112\Work	\\10.33.174.74\public\forMESC\FV113\Work	\\10.33.174.74\public\forMESC\FV114\Work	\\10.33.174.74\public\forMESC\FV115\Work
Backup Folder Path								
File Type Expected	FV101	FV103	FV106	FV111	FV112	FV113	FV114	FV115
CSV File Header Json	["Date","BufferID","EquipmentID","ModelNo","RecipeNo",	[{"RowStartIndex":0,"RowEndIndex":0,"Headers":["TrayNo	["Date","BufferID","EquipmentID","ModelNo","RecipeNo",	["Date","BufferID","EquipmentID","ModelNo","RecipeNo",	["Date","BufferID","EquipmentID","ModelNo","RecipeNo",	["Date","BufferID","EquipmentID","ModelNo","RecipeNo",	["Date","BufferID","EquipmentID","ModelNo","RecipeNo",	["Date","BufferID","EquipmentID","ModelNo","RecipeNo",

"DefectCode","ProcessSkipMode","ForcedOkMode","JellyRollID","CanID","AssemblyLotNo","TrayNo","PositionInTray","Classification"]

","ModelName","AssemblyLotNo","LineID","EquipmentID","StageNo","ProcessFlowNo","ProcessNo","ProductionRecipeNo","Classification","StartDate","EndDate","NumberOfTraysStacked","TrayPosition","StackingSerialNumber","ShelfNumber","TrayType"]},{"RowStartIndex":1,"RowEndIndex":-2,"Headers":["WindingID","CanID","CellID","Judgment"]},{"RowStartIndex":-1,"RowEndIndex":-1,"Headers":["Termin

"BadCode","ProcessSkipMode","ForcedOKMode","JellyRollID","CanID","CurrentLotCapInsulatorF1_TrayNO","CurrentLotCapInsulationPlateF1_MaterialLotName","PreviousLotCapInsulatorF1_TrayNO","PreviousLotCapInsulationPlateF1_MaterialLotName","CurrentLotAnodeCapF1_TrayNO","CurrentLotAnodeCapF1_MaterialLotName","PreviousLotAnodeCapF1_TrayNO","PreviousLotAnodeCapF1_MaterialLotName","CapInsulator

"DefectCode","ProcessSkipMode","ForcedOkMode","JellyRollID","CanID"]

"DefectCode","ProcessSkipMode","ForcedOkMode","JellyRollID","CanID","CellID"]

"DefectCode","ProcessSkipMode","ForcedOkMode","JellyRollID","CanID","CellID"]

"DefectCode","ProcessSkipMode","ForcedOkMode","JellyRollID","CanID","CellID"]

"DefectCode","ProcessSkipMode","ForcedOkMode","JellyRollID","CanID","CellID"]

		alText"]}]	F1_MaterialCode","AnodeCapF1_MaterialCode","CurrentLotCapInsulatorF2_TrayN","CurrentLotCapInsulationPlateF2_MaterialLotName","PreviousLotCapInsulatorF2_TrayNO","PreviousLotCapInsulationPlateF2_MaterialLotName","CurrentLotAnodeCapF2_TrayNO","CurrentLotAnodeCapF2_MaterialLotName","PreviousLotAnodeCapF2_TrayNO","PreviousLotAnodeCapF2_MaterialLotName","CapInsulatorF2_MaterialCode","Anod					

			eCapF2_MaterialCode","DrivingHistoryInformation","IDReadingHeadNo","AssemblySpindleSpindleNo","WeldingSpindleSpindleNo","IRSpindleSpindleNo","Pre-weldImageInspectionFeederSpindleNo","ImageInspectionFeederAfterWeldingSpindleNo","Pre-weldImageInspection1_DetailOfInspectionDecision","Dent","Scratch","Scratches_CrimpingAreaScratches","Scratches_CapPartScratches","Pre-weldImageInspe					

			ction1_P relimina ry","Pre- weldIma geInspe ction1_P relimina ry","Pre- weldIma geInspe ction1_P relimina ry","Pre- weldIma geInspe ction1_P relimina ry","Pre- weldIma geInspe ction1_P relimina ry","Pre- weldIma geInspe ction2_ DetailOf Inspecti onDecisi on","Lea kage"," Dirt","W eldLeak age","Cr impingA reaRust" ,"Crimpi ngAreaL eakage" ,"Pre- weldIma geInspe ction2_P relimina ry","Pre- weldIma geInspe ction2_P relimina ry","Pre- weldIma geInspe					

			ction2_Prelimina ry","Pre-weldIma geInspe ction2_Prelimina ry","Pre-weldIma geInspe ction2_Prelimina ry","Inn erFocalL ength","OuterFo calLengt h","With in_Pushi ngDista nce","La serHead Number ","LDDD ecision", "LDDRes erve","L DDRese rve","W aveform Evaluati onMachi neUseN umber", "Wavefo rmJudgi ngMachi neMax", "Wavefo rmJudgi ngMachi neMin","Waveform JudgingMachineAve","WaveformJudgingMachineReserve","WaveformJu					

dgingMachineReserve","WaveformJudgingMachineReserve","Post-weldImageInspection_DetailOfInspectionDecision","WeldWidth","WeldingDiameter","WeldLengthMax","WeldLengthMin","WeldingPosition","ImageInspectionAfterWelding_Preliminary","ImageInspectionAfterWelding_Preliminary","ImageInspectionAfterWelding_Preliminary","ImageInspectionAfterWelding_Preliminary","ImageInspectionAfterWelding_Preliminary

","IRInsp
ectionTe
sterNo",
"IRInspe
ctionSta
rtDateTi
me","IRI
nspectio
nResista
nceR","I
RTestRe
actance
ValueX",
"IRInspe
ctionOC
VMeasur
ements
V","IRIn
spection
Prelimin
ary","IRI
nspectio
nPrelimi
nar","Sp
indleWel
dingFreq
uency","
Number
OfBarna
cleWeldi
ng","Nu
mberOf
Welding
HeadWe
lds","Nu
mberOf
TimesPr
otective
GlassIsU
sed","IRI
nspectio
n_Numb
erOfTim
esTheAn
odeProb
eIsUsed
","IRInsp
ection_N
umberO
fCathod
eProbes
Used","

			Reserve","Rese					

			rve","Reserve","Reserve","Reserve","Reserve","Reserve","Reserve","Reserve","Reserve"]					
MES API Header Json			["Date","EquipmentID","ModelNo","DefectCode","CanID","CapInsulatingPlatePartNumber","CapInsulatorPartNumber","AnodeCapPartNumber1","AnodeCapPartNumber2"]	["Date","EquipmentID","ModelNo","DefectCode","CanID"]	["Date","EquipmentID","ModelNo","DefectCode","CanID","CellID"]	["Date","EquipmentID","ModelNo","DefectCode","CanID","CellID"]	["Date","EquipmentID","ModelNo","DefectCode","CanID","CellID"]	["Date","EquipmentID","ModelNo","DefectCode","CanID","CellID"]
Date & Time	Date	StartDate	Date	Date	Date	Date	Date	Date
Machine ID Column	EquipmentID	EquipmentID	EquipmentID	EquipmentID	EquipmentID	EquipmentID	EquipmentID	EquipmentID
Message Type Char Count	5	5	5	5	5	5	5	5
Work Flow Definition ID	ec7eb325-058e-495a-942c-bc508a6167c1	e0fd280f-9e07-48e0-9377-1459e2308349	e879aa53-6940-4dea-9e2d-	e879aa53-6940-4dea-9e2d-	e879aa53-6940-4dea-9e2d-	e879aa53-6940-4dea-9e2d-	e879aa53-6940-4dea-9e2d-	e879aa53-6940-4dea-9e2d-

			d9eb46765017	d9eb46765017	d9eb46765017	d9eb46765017	d9eb46765017	d9eb46765017
Work Flow Definition Name	VI Formation Based	VI Tray Based	VI Can Based	VI Can Based	VI Can Based	VI Can Based	VI Can Based	VI Can Based
Work Flow Version	7	22	11	11	11	11	11	11
Work Flow Category ID	1	1	1	1	1	1	1	1
Work Flow Reference	panasonic.mes.vi.adapter-vi	panasonic.mes.vi.adapter-vi	panasonic.mes.vi.adapter-vi	panasonic.mes.vi.adapter-vi	panasonic.mes.vi.adapter-vi	panasonic.mes.vi.adapter-vi	panasonic.mes.vi.adapter-vi	panasonic.mes.vi.adapter-vi
Step Name	Formation Common Signal Waiter	VI Start Signal Waiter	VI Common Signal Waiter	VI Common Signal Waiter	VI Common Signal Waiter	VI Common Signal Waiter	VI Common Signal Waiter	VI Common Signal Waiter
Work Flow Event Name	FormationComplete	VIStart	VIComplete	VIComplete	VIComplete	VIComplete	VIComplete	VIComplete
Tenant ID	1	1	1	1	1	1	1	1
Object Type	Material Group	Material Group	Material	Material	Material	Material	Material	Material
Druid Kafka Server	sm-main-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092	sm-main-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092	sm-main-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092	sm-main-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092	sm-main-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092	sm-main-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092	sm-main-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092	sm-main-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092
Druid Topic Name	panasonic.trace.topic	panasonic.trace.topic	panasonic.trace.topic	panasonic.trace.topic	panasonic.trace.topic	panasonic.trace.topic	panasonic.trace.topic	panasonic.trace.topic
Druid Configu	true	true	true	true	true	true	true	true

red								
Kafka UserName	kafka	kafka	kafka	kafka	kafka	kafka	kafka	kafka
Kafka PassWord	kafka	kafka	kafka	kafka	kafka	kafka	kafka	kafka
Kafka Authentication Type	plaintext	plaintext	plaintext	plaintext	plaintext	plaintext	plaintext	plaintext
Druid Kafka Group	cg_wf_dev_vi_fv101	cg_wf_dev_vi_fv103	cg_wf_dev_vi_fv106	cg_wf_dev_vi_fv111	cg_wf_dev_vi_fv112	cg_wf_dev_vi_fv113	cg_wf_dev_vi_fv114	cg_wf_dev_vi_fv115
Flow Start Kafka Connection	sm-main-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092	sm-main-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092	sm-main-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092	sm-main-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092	sm-main-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092	sm-main-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092	sm-main-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092	sm-main-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092
Flow Start Kafka Topic	panasonic.workflow.runtime.waitforevent.topic	panasonic.workflow.runtime.waitforevent.topic	panasonic.workflow.runtime.waitforevent.topic	panasonic.workflow.runtime.waitforevent.topic	panasonic.workflow.runtime.waitforevent.topic	panasonic.workflow.runtime.waitforevent.topic	panasonic.workflow.runtime.waitforevent.topic	panasonic.workflow.runtime.waitforevent.topic
Flow Start Kafka Auth	plaintext	plaintext	plaintext	plaintext	plaintext	plaintext	plaintext	plaintext
Flow Start Kafka UserName	kafka	kafka	kafka	kafka	kafka	kafka	kafka	kafka
Flow Start Kafka PassWord	kafka	kafka	kafka	kafka	kafka	kafka	kafka	kafka
Elastic Search	http://elastic:kW	http://elastic:kW	http://elastic:kW	http://elastic:kW	http://elastic:kW	http://elastic:kW	http://elastic:kW	http://elastic:kW

URL	1T74pcH5V715vP5Ad42Afs@sm-main-elasticsearch-es-http.sm-elasticsearch.svc.cluster.local:9200	1T74pcH5V715vP5Ad42Afs@sm-main-elasticsearch-es-http.sm-elasticsearch.svc.cluster.local:9200	1T74pcH5V715vP5Ad42Afs@sm-main-elasticsearch-es-http.sm-elasticsearch.svc.cluster.local:9200	1T74pcH5V715vP5Ad42Afs@sm-main-elasticsearch-es-http.sm-elasticsearch.svc.cluster.local:9200	1T74pcH5V715vP5Ad42Afs@sm-main-elasticsearch-es-http.sm-elasticsearch.svc.cluster.local:9200	1T74pcH5V715vP5Ad42Afs@sm-main-elasticsearch-es-http.sm-elasticsearch.svc.cluster.local:9200	1T74pcH5V715vP5Ad42Afs@sm-main-elasticsearch-es-http.sm-elasticsearch.svc.cluster.local:9200	1T74pcH5V715vP5Ad42Afs@sm-main-elasticsearch-es-http.sm-elasticsearch.svc.cluster.local:9200
Data Collections	[{"DataField":"ModelNo","DisplayName":"VI-CanRemoval-FV101-ModelNo","UnitOfMeasure":"Custom"},{"DataField":"RecipeNo","DisplayName":"VI-CanRemoval-FV101-RecipeNo","UnitOfMeasure":"Custom"},{"DataField":"ProcessSkipMode","DisplayName":"VI-CanRemoval-FV101-	{"CanDataCollection":[{"DataField":"ModelName","DisplayName":"VI-TrayBased-FV103-ModelName","UnitOfMeasure":"Custom"},{"DataField":"LineID","DisplayName":"VI-TrayBased-FV103-LineID","UnitOfMeasure":"Custom"},{"DataField":"StageNo","DisplayName":"VI-TrayBas	[{"DataField":"ModelNo","DisplayName":"VI-CanBased-FV106-ModelNo","UnitOfMeasure":"Custom"},{"DataField":"RecipeNo","DisplayName":"VI-CanBased-FV106-RecipeNo","UnitOfMeasure":"Custom"},{"DataField":"ProcessSkipMode","DisplayName":"VI-CanBased-FV106-	[{"DataField":"ModelNo","DisplayName":"VI-CanBased-FV111-ModelNo","UnitOfMeasure":"Custom"},{"DataField":"RecipeNo","DisplayName":"VI-CanBased-FV111-RecipeNo","UnitOfMeasure":"Custom"},{"DataField":"ProcessSkipMode","DisplayName":"VI-CanBased-FV111-	[{"DataField":"ModelNo","DisplayName":"VI-CanBased-FV112-ModelNo","UnitOfMeasure":"Custom"},{"DataField":"RecipeNo","DisplayName":"VI-CanBased-FV112-RecipeNo","UnitOfMeasure":"Custom"},{"DataField":"ProcessSkipMode","DisplayName":"VI-CanBased-FV112-	[{"DataField":"ModelNo","DisplayName":"VI-CanBased-FV113-ModelNo","UnitOfMeasure":"Custom"},{"DataField":"RecipeNo","DisplayName":"VI-CanBased-FV113-RecipeNo","UnitOfMeasure":"Custom"},{"DataField":"ProcessSkipMode","DisplayName":"VI-CanBased-FV113-	[{"DataField":"ModelNo","DisplayName":"VI-CanBased-FV114-ModelNo","UnitOfMeasure":"Custom"},{"DataField":"RecipeNo","DisplayName":"VI-CanBased-FV114-RecipeNo","UnitOfMeasure":"Custom"},{"DataField":"ProcessSkipMode","DisplayName":"VI-CanBased-FV114-	[{"DataField":"ModelNo","DisplayName":"VI-CanBased-FV115-ModelNo","UnitOfMeasure":"Custom"},{"DataField":"RecipeNo","DisplayName":"VI-CanBased-FV115-RecipeNo","UnitOfMeasure":"Custom"},{"DataField":"ProcessSkipMode","DisplayName":"VI-CanBased-FV115-

	Process SkipMode","UnitOfMeasure":"Custom","DataType":"Number"},{"DataField":"ForcedOkMode","DisplayName":"VI-CanRemoval-FV101-ForcedOkMode","UnitOfMeasure":"Custom","DataType":"Number"},{"DataField":"Classification","DisplayName":"VI-CanRemoval-FV101-Classification","UnitOfMeasure":"Custom","DataType":"Number"}]	ed-FV103-StageNo","UnitOfMeasure":"Custom","DataType":"Number"},{"DataField":"ProcessFlowNo","DisplayName":"VI-TrayBased-FV103-ProcessFlowNo","UnitOfMeasure":"Custom","DataType":"Number"},{"DataField":"ProcessNo","DisplayName":"VI-TrayBased-FV103-ProcessNo","UnitOfMeasure":"Custom","DataType":"Number"},{"DataField":"ProductionRecipeNo","DisplayName":"VI-	Process SkipMode","UnitOfMeasure":"Custom","DataType":"Number"},{"DataField":"ForcedOkMode","DisplayName":"VI-CanBased-FV106-ForcedOkMode","UnitOfMeasure":"Custom","DataType":"Number"},{"DataField":"TrayIDCurrentLotOfCapInsulator1","DisplayName":"VI-CanBased-FV106-TrayIDCurrentLotOfCapInsulator1","UnitOfMeasure":"Custom"},{"DataField":"MaterialLotNameCu	Process SkipMode","UnitOfMeasure":"Custom","DataType":"Number"},{"DataField":"ForcedOkMode","DisplayName":"VI-CanBased-FV111-ForcedOkMode","UnitOfMeasure":"Custom","DataType":"Number"}]	Process SkipMode","UnitOfMeasure":"Custom","DataType":"Number"},{"DataField":"ForcedOkMode","DisplayName":"VI-CanBased-FV112-ForcedOkMode","UnitOfMeasure":"Custom","DataType":"Number"}]	Process SkipMode","UnitOfMeasure":"Custom","DataType":"Number"},{"DataField":"ForcedOkMode","DisplayName":"VI-CanBased-FV113-ForcedOkMode","UnitOfMeasure":"Custom","DataType":"Number"}]	Process SkipMode","UnitOfMeasure":"Custom","DataType":"Number"},{"DataField":"ForcedOkMode","DisplayName":"VI-CanBased-FV114-ForcedOkMode","UnitOfMeasure":"Custom","DataType":"Number"}]	Process SkipMode","UnitOfMeasure":"Custom","DataType":"Number"},{"DataField":"ForcedOkMode","DisplayName":"VI-CanBased-FV115-ForcedOkMode","UnitOfMeasure":"Custom","DataType":"Number"}]

		TrayBased-FV103-ProductionRecipeNo","UnitOfMeasure":"Custom"},{"DataField":"Classification","DisplayName":"VI-TrayBased-FV103-Classification","UnitOfMeasure":"Custom"}],"TrayDataCollection":[{"DataField":"EndDate","DisplayName":"ATRR-TRAY-UPDATED-DATETIME","UnitOfMeasure":"Custom"},{"DataField":"NumberOfTraysStacked","DisplayName":"ATRR-TRAY-	rrentLotOfCapInsulator1","DisplayName":"VI-CanBased-FV106-MaterialLotNameCurrentLotOfCapInsulator1","UnitOfMeasure":"Custom"},{"DataField":"TrayIDPreviousLotOfCapInsulator1","DisplayName":"VI-CanBased-FV106-TrayIDPreviousLotOfCapInsulator1","UnitOfMeasure":"Custom"},{"DataField":"MaterialLotNamePreviousLotOfCapInsulatingPlate1","DisplayName":"VI-CanBased-					

		STACKING-COUNT","UnitOfMeasure":"Custom","DataType":"Number"}, {"DataField":"TrayPosition","DisplayName":"ATRR-TRAY-POSITION","UnitOfMeasure":"Custom","DataType":"Number"}, {"DataField":"StackingSerialNumber","DisplayName":"ATRR-TRAY-STACKING-SERIAL-NUM","UnitOfMeasure":"Custom","DataType":"Number"}, {"DataField":"ShelfNumber","DisplayName":"ATRR-TRAY-SHELF-	FV106-MaterialLotNamePreviousLotOfCapInsulatingPlate1","UnitOfMeasure":"Custom"}, {"DataField":"TrayIDCurrentLotOfCapInsulatingPlate2","DisplayName":"VI-CanBased-FV106-TrayIDCurrentLotOfCapInsulatingPlate2","UnitOfMeasure":"Custom"}, {"DataField":"MaterialLotNameCurrentLotOfCapInsulatingPlate2","DisplayName":"VI-CanBased-FV106-MaterialLotNameCurrentLotOfCapInsulatingPlat					

		NUM","UnitOfMeasure":"Custom"},{"DataField":"TrayType","DisplayName":"ATRR-TRAY-TYPE","UnitOfMeasure":"Custom"}]}	e2","UnitOfMeasure":"Custom"},{"DataField":"TrayIDPreviousLotOfCapInsulatingPlate2","DisplayName":"VI-CanBased-FV106-TrayIDPreviousLotOfCapInsulatingPlate2","UnitOfMeasure":"Custom"},{"DataField":"MaterialLotNamePreviousLotOfCapInsulationPlate2","DisplayName":"VI-CanBased-FV106-MaterialLotNamePreviousLotOfCapInsulationPlate2","UnitOfMeasure":"Custom"},{"DataField":"Tra					

yIDCurrentLotOfAnodeCap1","DisplayName":"VI-CanBased-FV106-TrayIDCurrentLotOfAnodeCap1","UnitOfMeasure":"Custom"},{"DataField":"MaterialLotNameCurrentLotOfAnodeCap1","DisplayName":"VI-CanBased-FV106-MaterialLotNameCurrentLotOfAnodeCap1","UnitOfMeasure":"Custom"},{"DataField":"TrayIDPreviousLotOfAnodeCap1","DisplayName":"VI-CanBased-FV106-TrayIDPreviousL

otOfAnodeCap1","UnitOfMeasure":"Custom"},{"DataField":"MaterialLotNamePreviousLotOfAnodeCap1","DisplayName":"VI-CanBased-FV106-MaterialLotNamePreviousLotOfAnodeCap1","UnitOfMeasure":"Custom"},{"DataField":"TrayIDCurrentLotOfAnodeCap2","DisplayName":"VI-CanBased-FV106-TrayIDCurrentLotOfAnodeCap2","UnitOfMeasure":"Custom"},{"DataField":"MaterialLotNameCurrentLot

OfAnodeCap2","DisplayName":"VI-CanBased-FV106-MaterialLotNameCurrentLotOfAnodeCap2","UnitOfMeasure":"Custom"},{"DataField":"TrayIDPreviousLotOfAnodeCap2","DisplayName":"VI-CanBased-FV106-TrayIDPreviousLotOfAnodeCap2","UnitOfMeasure":"Custom"},{"DataField":"MaterialLotNamePreviousLotOfAnodeCap2","DisplayName":"VI-CanBased-FV106-MaterialLotNamePreviou

sLotOfAnodeCap2","UnitOfMeasure":"Custom"},{"DataField":"CapInsulatorPartNumber","DisplayName":"VI-CanBased-FV106-CapInsulatorPartNumber","UnitOfMeasure":"Custom"},{"DataField":"CapInsulatingPlatePartNumber","DisplayName":"VI-CanBased-FV106-CapInsulatingPlatePartNumber","UnitOfMeasure":"Custom"},{"DataField":"AnodeCapPartNumber1","DisplayName":"VI-CanBased-

			FV106-AnodeCapPartNumber1","UnitOfMeasure":"Custom"}, {"DataField":"AnodeCapPartNumber2","DisplayName":"VI-CanBased-FV106-AnodeCapPartNumber2","UnitOfMeasure":"Custom"}]					
Adapter Setting	{"ProductSystemVersion":"1","ProductUserVersion":"1","ProcessId":"FV101","ProcessName":"VI-START","ProcessSegmentName":"FV101","ProcessSegmentStatus":"FV101"}	{"ProductSystemVersion":"1","ProductUserVersion":"1","ProcessId":"FV103","ProcessName":"VI10","ProcessSegmentName":"FV103","ProcessSegmentStatus":"FV103"}	{"ProductSystemVersion":"3","ProductUserVersion":"1","ProcessId":"FV106","ProcessName":"VI","ProcessSegmentName":"FV106","ProcessSegmentStatus":"FV106","LocationInfo":{"LineLocation":{"Type":1,"Label	{"ProductSystemVersion":"1","ProductUserVersion":"1","ProcessId":"FV111","ProcessName":"VI","ProcessSegmentName":"FV111","ProcessSegmentStatus":"FV111"}	{"ProductSystemVersion":"1","ProductUserVersion":"1","ProcessId":"FV112","ProcessName":"VI","ProcessSegmentName":"FV112","ProcessSegmentStatus":"FV112"}	{"ProductSystemVersion":"1","ProductUserVersion":"1","ProcessId":"FV113","ProcessName":"VI","ProcessSegmentName":"FV113","ProcessSegmentStatus":"FV113"}	{"ProductSystemVersion":"1","ProductUserVersion":"1","ProcessId":"FV114","ProcessName":"VI","ProcessSegmentName":"FV114","ProcessSegmentStatus":"FV114"}	{"ProductSystemVersion":"1","ProductUserVersion":"1","ProcessId":"FV115","ProcessName":"VI","ProcessSegmentName":"FV115","ProcessSegmentStatus":"FV115"}

			":"LINE"},"ConsumeLocation":{"Type":1,"Label":"RAW_MATERIAL"}}}					
Implicit Start	true	true	true	true	true	true	true	true
CAN Work Flow Definition ID	NA	39ed1d09-a09c-4b19-865f-677ce6a6eedf	NA	NA	NA	NA	NA	NA
CAN Work Flow Definition Name	NA	VI Can Based	NA	NA	NA	NA	NA	NA
CAN Work Flow Version	NA	1	NA	NA	NA	NA	NA	NA
CAN Work Flow Step Name	NA	VI Common Signal Waiter	NA	NA	NA	NA	NA	NA
CAN Work Flow Event Name	NA	VIComplete	NA	NA	NA	NA	NA	NA
CAN Work Flow Implicit Start	NA	FALSE	NA	NA	NA	NA	NA	NA

IF Connection Information for MEBD Prod

Name	Value
Elastic Search URL	http://elastic:kW1T74pcH5V715vP5Ad42Afs@sm-main-elasticsearch-es-http.sm-elasticsearch.svc.cluster.local:9200
Authentication URL	https://panasonic-user-authentication-coreapp-api-keycloak-internal-svc.smpum.svc.cluster.local/realms/Default/protocol/openid-connect/token
Authentication user	admin
Authentication password	**********
Source folder path	{{\\10.33.161.92\prod_connector_share\}}{{contractname}}
Source Username (SFTP Username)	User01
Source Password	*******
Error Folder Path	{{\\10.33.161.92\prod_connector_share\}}{{contractname}}\error
Working Folder Path	{{\\10.33.161.92\prod_connector_share\}}{{contractname}}\work
Backup Folder Path	{{\\10.33.161.92\prod_connector_share\}}{{contractname}}\backup
Workflow Definition ID	
Workflow Version	
API URL	http://panasonic-inventory-runtime-api-service.smpim.svc.cluster.local/pitchProcessing
Carrier URL Path	http://panasonic-inventory-runtime-api-service.smpim.svc.cluster.local/carriers/searchContents
Workflow URL	http://panasonic-workflowmanagement-runtime-api-service.smpwf.svc.cluster.local/orchestrator/signal
Kafka server	sm-main-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092
Adapter Setting	[{"Process":"ProcessCompletion","Url":"https://mebd.wakayama.mes.energy.jp.panasonic.com/if/connections/validate-process-completion-1/start"}]}}

MES Adapter Configuration

Color	**Significance of color in the table**
🟦	These fields are environment specific. Meaning it will be configured differently for SFDEV, MEBD-QA, MEBD-PROD, etc
◼	These fields are again environment specific. But these fields also need to be updated whenever you update or perform version-up in the respective coreapp. For example Workflow version needs to be updated if we upgrade the version. Similarly for Product System Version.
🟨	This fields are according to MEBD PROD environment

Configuration Type	**Adapter Information**	**Adapter Connection**											
		FE111	**FE112**	**FE114**	**FE115**	**FE123**	**FE124**	**FE126**	**FE127**	**FE129**	**FE130**	**FE132**	**FE133**
	Adapter & Version	**panasonic-generic-csv-cifs-watch-adapter & 3.0.1**	**panasonic-generic-csv-cifs-watch-adapter & 3.0.1**	**panasonic-generic-csv-cifs-watch-adapter & 3.0.1**	**panasonic-generic-csv-cifs-watch-adapter & 3.0.1**	**panasonic-generic-csv-cifs-watch-adapter & 3.0.1**	**panasonic-generic-csv-cifs-watch-adapter & 3.0.1**	**panasonic-generic-csv-cifs-watch-adapter & 3.0.1**	**panasonic-generic-csv-cifs-watch-adapter & 3.0.1**	**panasonic-generic-csv-cifs-watch-adapter & 3.0.1**	**panasonic-generic-csv-cifs-watch-adapter & 3.0.1**	**panasonic-generic-csv-cifs-watch-adapter & 3.0.1**	**panasonic-generic-csv-cifs-watch-adapter & 3.0.1**

	Configuration Field												
Generic	**Adapter Name**	edgepc-174-71-fe111											
	Source Folder Path	\\10.33.174.71\public\forMESC\FE111											
	Source Folder Username	smb01											
	Source Folder Password	WakaKino01											
	File Extension Filter	["csv","CSV"]											

Ignore File Extensions	["csv_", "CSV_"]											
Error Folder Path	\\10.33.174.71\public\forMESC\FE111\Error											
Working Folder Path	\\10.33.174.71\public\forMESC\FE111\Work											
Backup Folder Path	\\10.33.174.71\public\forMESC\FE111\Backup											
CSV File Header Json	["DateTime","TriggerWord","EquipmentId","RecipeNum	["DateTime","TriggerWord","EquipmentId",	["DateTime","TriggerWord","EquipmentId","Re	["DateTime","TriggerWord","EquipmentId", "Rec	["DateTime","TriggerWord","EquipmentId","Re	["DateTime","TriggerWord","EquipmentIdentl	["DateTime","TriggerWord","EquipmentId","Re	["DateTime","TriggerWord","EquipmentId","RecipeNu	["DateTime","TriggerWord","EquipmentId","Recipe	["DateTime","TriggerWord","EquipmentId","Re	["DateTime","TriggerWord","EquipmentId","R	["DateTime","TriggerWord","EquipmentId","RecipeNumber","

ber", "Mod elCo de","Userl d","P rodu ction LotO dd","P rodu ctio nLot Even ","Foi lInpu tQua ntity ","Foi lPart Num ber", "FoilL ot","P rodu ctio nDat e","S lurry Tank Num ","St orag eTan kNu mber ","Un windi ngAx is","R emai ning Diam eter" ,"Qu ality Data ","Qu ality Data 1","Q ualit

"Re cipe Nu mbe r","P rodu ctio nLot ","Pr oduc tionL ot2" ,"Foi lLot ","Pr odu ctio nda y"," Tier ASt ora geT ank Nu mbe r","T ierB Stor age Tan kNu mbe r"," Unw indi ngT urre t"," Rem aini ngD iam eter OfU nwi

cipe Num ber", "Mo delN o"," User ID"," Prod uctio nLot ","Pr oduc tionL ot2", "Foil Lot", "Pro duct iond ay"," Tier ASto rage Tank Num ber", "Tier BSto rage Tank Num ber", "Un wind ingT urret ","R ema ining Dia met erOf Unwi ndin gSpli ce"," FoilP artN umb er"," Qual

ipeN umb er","P rodu ctio nLot ","Pr oduc tionL ot"," Colu mnN um", "Out putS heet Qua ntity ","G oodP itch Cou nt"," NGPi tchC ount ","M eter sPro duce d","P rodu ction day" ,"Un wind ingC ount er"," AWei ghto fslur ryAp plied ","B Slurr yAp plica tion Weig ht","

cipe Num ber" ,"Mo delC ode" ,"Us erID ","Pr oduc tionL ot"," Colu mnN um", "Out putL ot"]

d","R eci peN um ber" ,"M odel Cod e","U serI d"," Use rId", "Pro duc tion Lot" ,"Un win din gLo tNa me" ,"Pr odu ced Pitc h"," Bad Pitc h"," Pro duc edG ood s"," Pro duc tion Day ","M axi mu mR ollin gSp eed ","P ath Cou nt", "Sta rtTe nsio

cipe Num ber "," Mod elCod e ","U serI d"," Prod ucti onLo t","I nput Lot"]

mber ","M odel No","U serI D","P rodu ction Lot", "Inpu tLot" ,"Ha ndle dShe etQu antit y","O utpu tShe etQu antit y","G oodP itchC ount ","N GPitc hCou nt"," Mete rsPro duce d","P rodu ction Day" ,"Un windi ngCo unte r","M axim umR ollin gSpe ed"," Path Coun t","St artin gTen sion1

Num ber", "Mod elCo de"," Prod uctio nLot ","In putL ot"]

cipe Num ber" ,"Mo delC ode" ,"Us erId" ,"Pro duct ionL ot"," Unw indi ngL otNa me", "Pro duc edS heet Cou ntEn try", "Pro duc edS heet Cou ntEx it"," Prod uce dGo ods" ,"NG Pitc hes" ,"Pro duc edG oods Met ers", "Pro duct ionP erfor man ceD ay", "Em

ecip eNu mbe r"," Mod elCo de", "Us erID ","Pr odu ctio nLot ","In putL ot"]

Model Code" ,"User Id","P roduc tionLo t","Un windi ngLot Name ","Ha ndled Sheet Quant ity"," Outpu tShee tQuan tity"," Good Pitchc ount", "NGPi tchCo unt"," Meter sProd uced" ,"Prod uction Date" ,"Unw indco unter ","Em ptyva cant1 ","Roll temp eratur e1str ollcen ter"," Rollte mper ature 2ndro llcent er","R ollte mper ature

yDat a2"," Quali tyDa ta3", "Qua lityD ata4" ,"Qu ality Data 5","Q ualit yDat a6"," Quali tyDa ta7", "Qua lityD ata8"] ndin gSpl ice", "Foil Part Nu mbe r"," Qua lity Dat a1", "Qu ality Dat a2", "Qu ality Dat a3", "Qu ality Dat a4"] ityD ata1 ","Q ualit yDat a2", "Qu ality Data 3"," Qual ityD ata4 "] DieH ead Man age men tNoF ront" ,"Die Hea dMa nage men tNoB ack", "Ser voSp eedS ettin g","E mpt yVac ant", "Sim Boar dMa nage men tNo" ,"Shi mPla teTh ickn ess", "Shi mPla teOp enin gWid th"," Coat ingS peed ","Ex haus tAirfl ow", "Tota lLen gthI nter mitt entPi nPa ss1" ,"En dTe nsio nPa ss1" ,"Ta per Star tDia met erP ass 1"," Star tTen sion Pas s2", "En dTe nsio nPa ss2" ,"Ta per Star tDia met erP ass 2"," Star tTen sion Pas s3", "En dTe nsio nPa ss3" ,"Ta per Star tDia met erP ass 3"," Ben stpas s","E ndTe nsion 1stp ass", "Tap erSta rtDia mete r1stp ass", "Star tingT ensio n2nd Pass" ,"End Tensi on2n dPas s","T aper Start Diam eter2 ndPa ss"," Start ingTe nsion 3rdP ass", "End Tensi on3r dPas s","T aper Start Diam eter3 rdPa ss"," Bend Press ureS ettin g1st Pass" ,"Be ndPr pty/ Vaca nt1" ,"Rol lTem pera ture 1stR oll"," Emp ty/V acan t2"," Emp ty/V acan t3"," Roll Tem pera ture 2nd Roll" ,"Em pty/ Vaca nt4" ,"Em pty/ Vaca nt5" ,"Rol lTem pera ture 3rdR oll"," Emp ty/V acan t6"," Emp ty/V acan t7"," Roll Tem pera ture 4thR oll"," Emp 3rdrol lcente r","Ro lltem perat ure4t hrollc enter ","Roll temp eratur e5thr ollcen ter"," Rollte mper ature 6throl lcente r","Ro lltem perat ure7t hrollc enter ","Roll temp eratur e8thr ollcen ter"," Rollte mper ature 9throl lcente r","Ro lltem perat ure10 throll cente r","Ro lltem perat ure11 throll cente r","Ro lltem perat ure12

tchS dPr essu ty/V throll
etVal ess reset acan cente
ue"," ure ting2 t8"," r","Ro
Tabl Sett ndPa Tens llItem
eCo ingP ss"," ionS perat
atin ass Bend ettin ure13
gLen 1"," Press gVal throll
gthS Ben ures ueU cente
etVal dPr ettin nwin r","Ro
ue"," ess g3rd ding llItem
Back ure Pass" ","Te perat
Coat Sett ,"Sec nsio ure14
ingL ingP ondB nSet throll
engt ass endP poin cente
hSet 2"," ressu tAcc r","E
Valu Ben reSet umu mpty
e"," dPr ting1 late vacan
Hea ess stPas dFee t2","H
dToH ure s","S d"," ybrud
eadT Sett econ Tens RollTe
ailTo ingP dBen ionS mper
Tail", ass dPre etpo ature
"Nu 3"," ssure intIn 1","H
mbe Pres Setti Feed ybrud
rOfD sPre ng2n ","Te RollTe
efect ssur dPas nsio mper
iveS eSe s","S nSet ature
heet ttin econ poin 2","E
s"," gPa dBen tCoo mpty
wind ss1" dPre lingI vacan
ingT ,"Pr ssure nlet" t3","S
urret ess Setti ,"Te lowco
","1 Pres ng3r nsio olingr
CSer sur dPas nSet oll1","
voVa eSe s","P poin Slowc
lveC ttin ressP tOut ooling
ount gPa ressu Feed roll2",
","2 ss2" reSet ","Te "Slow
CSer ,"Pr ting1 nsio coolin
voVa ess stPas nSet groll3
lveC Pres s","P poin ","Slo
ount sur ressP tWin wcool
","Q eSe ressu ding ingroll
ualit ttin reSet Star 4","Sl
yDat gPa ting2 ting owco
a1"," ss3" ndPa Tens olingr
Qual ,"Pr ss"," ion", oll5","
ityD eloa Press "Ten Slowc
ata2 dPa Press sion ooling
","Q ss1" ureS SetV roll6",
ualit ,"Pr ettin alue "Slow
yDat eloa g3rd Win coolin

a3","QualityData4"]

dPass2","PreloadPass3","MaxOilPressureTemperature","CorrectionFactorAHeatingControl","CorrectionFactorACoolingControl","UpperRollOffsetTemperature","LowerRollOffsetTemperature"]

Pass","SecondPressPressureSetting1stPass","SecondPressPressureSetting2ndPass","SecondPressPressureSetting3rdPass","Preload1stPass","Preload2ndPass","Preload3rdPass","MaxOilPressureTemperature","SecondMaxoilPressureTemperature","Correctio

dingEndTension","TensionSetpointWindingTaperStartDiameter","Empty/Vacant9","Empty/Vacant10","LineSpeedSetpoint","QualityData"]

groll7","Slowcoolingroll8","Slowcolingroll9","Slowcooolingroll10","Slowcoolingroll11","Emptyvacant4","Tensionsettingvalueunwinding","Tensionsetpointaccumulatedfeed","Tensionsetpointinfeed","Tensionsetpointcoolinginlet","Tensionsetpointoutfeed","Tensionsetpointwinding Startingtension",

								nFactorHeatingControl","CorrectionFactorCoolingControl","UpperRollOffsetTemperature","LowerRollOffsetTemperature"]				"Tensionsetvaluewindingendtension","Tensionsetpointwinding Taperstartdiameter","Cooling watertemparature","Emptyvacant5","Linespeedsetpoint"]
Date & Time Column	DateTime	DateTime	DateTime	DateTime	DateTime	DateTime	DateTime	DateTime	DateTime	DateTime	DateTime	DateTime
Machine ID Column	EquipmentId	EquipmentId	EquipmentId	EquipmentId	EquipmentId	EquipmentId	EquipmentId	EquipmentId	EquipmentId	EquipmentId	EquipmentId	EquipmentId
Message Type Char Co	5											

unt												
DruidKafkaServer	sm-main-kafka-kafka-kafka-bootstrap.sm-kafka.svc.cluster.local:9092											
DruidTopicName	panasonic.electrode.topic											
KafkaUserName	kafka											
KafkaPassword	kafka											
KafkaAuthenticationType	plaintext											
Elastic	http://elastic:k											

	Search URL	W1T74pcH5V715vP5Ad42Afs@sm-main-elasticsearch-es-http.sm-elasticsearch.svc.cluster.local:9200											
	Publish Interval Milliseconds	2											
MES	**Adapter Information**	**Adapter Connection**											
		Coating - Cathode	Coating - Anode	Pressing - Cathode	Pressing - Anode	Sunburn - Cathode	Sunburn - Anode						
	Adapter & Ve	panasonic-mes-api-adapter v1.0.0+	panasonic-mes-api-adapter v1.0.0+	panasonic-mes-api-adapter v1.0.0+	panasonic-mes-api-adapter v1.0.0+	panasonic-mes-api-adapter v1.0.0+	panasonic-mes-api-adapter v1.0.0+						

rsion						
Configuration Field						
API Adapter Name	Coating-Cathode	Coating-Anode	Pressing-Cathode	Pressing-Anode	SB-Cathode	SB-Anode
Expected File IDs	FE111,FE112	FE114,FE115	FE123,FE124	FE126,FE127	FE129,FE130	FE132,FE133
Start URL	start	start	start	start	start	start
End URL	end	end	end	end	end	end
Work Flow Definition ID	3bd50244-0f23-44bf-a16a-f36cca9a9338	3bd50244-0f23-44bf-a16a-f36cca9a9338	8d9e5704-6cd3-456d-a2a9-0017015717d6	8d9e5704-6cd3-456d-a2a9-0017015717d6	04684be7-da6a-451f-b88c-5cb9c7ecb05c	04684be7-da6a-451f-b88c-5cb9c7ecb05c
Work Flow Definition Name	Coating Workflow	Coating Workflow	Pressing Workflow	Pressing Workflow	Sunburn Workflow	Sunburn Workflow
Work	1	1	1	1	1	1

Flow Version						
WorkFlow Reference	panasonic.mes.api.adapter-coating	panasonic.mes.api.adapter-coating	panasonic.mes.api.adapter-pressing	panasonic.mes.api.adapter-pressing	panasonic.mes.api.adapter-SB	panasonic.mes.api.adapter-SB
WorkFlow CategoryID	1	1	1	1	1	1
Product System Version	1	1	1	1	1	1
Product User Version	1	1	1	1	1	1
Process Name	CA-COATING	AN-COATING	CA-PRESSING	AN-PRESSING	CA-SB	AN-SB
StartProcessID	FE111	FE114	FE123	FE126	FE129	FE132

End Process ID	FE112	FE115	FE124	FE127	FE130	FE133
Work Flow Step Name	Coating Complete Signal Waiter	Coating Complete Signal Waiter	Pressing Complete Signal Waiter	Pressing Complete Signal Waiter	Sunburn Complete Signal Waiter	Sunburn Complete Signal Waiter
Work Flow Event Name	CoatingComplete	CoatingComplete	PressingComplete	PressingComplete	SunburnComplete	SunburnComplete
Quality Group Attribute Name	QualityGroupId	QualityGroupId	QualityGroupId	QualityGroupId	QualityGroupId	QualityGroupId
Data Collections for Workflow	[{"DataField":"RecipeNumber","DisplayName":"ModelNo","UnitOfMeasure":"Custom"},{"DataField":"FoilLot","DisplayName":"FoilLot","UnitOfMeasure":"Custom"},{"DataField":"FoilPartNumber","Di	[{"DataField":"RecipeNumber","DisplayName":"ModelNo","UnitOfMeasure":"Custom"},{"DataField":"FoilLot","DisplayName":"FoilLot","UnitOfMeasure":"Custom"},{"DataField":"FoilPartNumber","Di	[{ "DataField": "ModelCode", "DisplayName": "ModelNo", "UnitOfMeasure": "Custom" }, {	[{ "DataField": "ModelCode", "DisplayName": "ModelNo", "UnitOfMeasure": "Custom" }, { "DataField":	[{ "DataField": "ModelCode", "DisplayName": "ModelNo", "UnitOfMeasure": "Custom" }, {	[{ "DataField": "ModelCode", "DisplayName": "ModelNo", "UnitOfMeasure": "Custom" }, { "DataField":

	splayName":"FoilPartNumber","UnitOfMeasure":"Custom"},{"DataField":"ProductionDate","DisplayName":"ProductionDate","UnitOfMeasure":"Custom"},{"DataField":"SlurryTankNoA","DisplayName":"SlurryTankNoA","UnitOfMeasure":"Custom"},{"DataField":"SlurryTankNoB","DisplayName":"SlurryTankNoB","UnitOfMeasure":"Custom"},{"DataField":"OutputSheetQuantity","DisplayName":"ATRR-INV-STOCK-COUNT","UnitOfMeasure":"Custom"},{"DataField":"HandledQuantity","DisplayName":"ATRR-INV-HANDLED-COUNT","UnitOfMeasure":"Custom"},{"DataField	splayName":"FoilPartNumber","UnitOfMeasure":"Custom"},{"DataField":"ProductionDate","DisplayName":"ProductionDate","UnitOfMeasure":"Custom"},{"DataField":"SlurryTankNoA","DisplayName":"SlurryTankNoA","UnitOfMeasure":"Custom"},{"DataField":"SlurryTankNoB","DisplayName":"SlurryTankNoB","UnitOfMeasure":"Custom"},{"DataField":"OutputSheetQuantity","DisplayName":"ATRR-INV-STOCK-COUNT","UnitOfMeasure":"Custom"},{"DataField":"HandledQuantity","DisplayName":"ATRR-INV-HANDLED-COUNT","UnitOfMeasure":"Custom"},{"DataField	"DataField":"ProductionDate","DisplayName":"ProductionDate","UnitOfMeasure":"Custom"},{"DataField":"OutputSheetQuantity","DisplayName":"OutputSheetQuantity","UnitOfMeasure":"Custom"},{"DataField":"HandledQuantity","DisplayName":"ATRR-INV-HANDLED-COUNT","UnitOfMeasure":"Custom"},{"DataField	"ProductionDate","DisplayName":"ProductionDate","UnitOfMeasure":"Custom"},{"DataField":"OutputSheetQuantity","DisplayName":"OutputSheetQuantity","UnitOfMeasure":"Custom"},{"DataField":"HandledQuantity","DisplayName":"ATRR-INV-HANDLED-COUNT","UnitOfMeasure":"Custom"},{"DataField":"MeterProduced","DisplayName":"MeterProd	"ProductionDate","DisplayName":"ProductionDate","UnitOfMeasure":"Custom"},{"DataField":"OutputSheetQuantity","DisplayName":"OutputSheetQuantity","UnitOfMeasure":"Custom"},{"DataField":"HandledQuantity","DisplayName":"ATRR-INV-HANDLED-COUNT","UnitOfMeasure":"Custom"},{"DataField":"MeterProduced","DisplayNa	"ProductionDate","DisplayName":"ProductionDate","UnitOfMeasure":"Custom"},{"DataField":"OutputSheetQuantity","DisplayName":"OutputSheetQuantity","UnitOfMeasure":"Custom"},{"DataField":"HandledQuantity","DisplayName":"ATRR-INV-HANDLED-COUNT","UnitOfMeasure":"Custom"},{"DataField":"MeterProduced","DisplayName":"MeterProdu

	":"MeterProduced","DisplayName": "MeterProduced","UnitOfMeasure":"Custom"}]	":"MeterProduced","DisplayName": "MeterProduced","UnitOfMeasure":"Custom"}]	: "MeterProduced", "DisplayName": "MeterProduced", "UnitOfMeasure": "Custom" }]	uced", "UnitOfMeasure": "Custom" }]	me": "MeterProduced", "UnitOfMeasure": "Custom" }]	ced", "UnitOfMeasure": "Custom" }]
Start Timer - Copy Timer	false	false	true	true	true	true
Start Timer - Timer Tag	COATINGSTART	COATINGSTART	PRESSINGSTART	PRESSINGSTART	SBSTART	SBSTART
Start Timer - Aging WF DefinitionId	NA	NA	3bd50244-0f23-44bf-a16a-f36cca9a9338	3bd50244-0f23-44bf-a16a-f36cca9a9338	Aging WF DefinitionId	Aging WF DefinitionId
Start Timer - Aging Workflow Ver	NA	NA	1	1	1	1

sion						
End Timer - Copy Timer	false	false	false	false	false	false
End Timer - Timer Tag	COATINGEND	COATINGEND	PRESINGEND	PRESSINGEND	SBEND	SBEND
End Timer - Aging WF DefinitionId	Aging WF DefinitionId	Aging WF DefinitionId	3bd50244-0f23-44bf-a16a-f36cca9a9338	3bd50244-0f23-44bf-a16a-f36cca9a9338	Aging WF DefinitionId	Aging WF DefinitionId
End Timer - Aging Workflow Version	1	1	1	1	1	1
Propagation Process Details	{ "PropagationWorkflowDefinitionId": "b6c33e63-f2d4-427e-9c3e-cabcf14cd7	{ "PropagationWorkflowDefinitionId": "b6c33e63-f2d4-427e-9c3e-cabcf14cd7	{ "PropagationWorkflowDefinitionId": "b6c33e63-f2d4-427e-	{ "PropagationWorkflowDefinitionId": "b6c33e63-f2d4-427e-9c3e-cabcf14cd7	{ "PropagationWorkflowDefinitionId": "b6c33e63-f2d4-427e-9c3e-cabcf14cd7	{ "PropagationWorkflowDefinitionId": "b6c33e63-f2d4-427e-9c3e-cabcf14cd75

	5f", "PropagationWorkflowVersion": "1" }	5f", "PropagationWorkflowVersion": "1" }	9c3e-cabcf14cd75f", "PropagationWorkflowVersion": "1" }	5f", "PropagationWorkflowVersion": "1" }	5f", "PropagationWorkflowVersion": "1" }	f", "PropagationWorkflowVersion": "1" }

Process Completion Details	{
	"ProcessCompletionWorkflowUrls": [
	{
	"Process": "ProcessCompletion",
	"Url": "https://sfdev.panasonicfa.com/if/connections/validate-process-completion-1/start"
	}
	]
	}
		{
		"ProcessCompletionWorkflowUrls": [
		{
		"Process": "ProcessCompletion",
		"Url": "https://sfdev.panasonicfa.com/if/connections/validate-process-completion-1/start"
		}
		]
		}
			{
			"ProcessCompletionWorkflowUrls": [
			{
			"Process": "ProcessCompletion",
			"Url": "https://sfdev.panasonicfa.com/if/connections/validate-process-completion-1/start"
			}
			]
			}
				{
				"ProcessCompletionWorkflowUrls": [
				{
				"Process": "ProcessCompletion",
				"Url": "https://sfdev.panasonicfa.com/if/connections/validate-process-completion-1/start"
				}
				]
				}
					{
					"ProcessCompletionWorkflowUrls": [
					{
					"Process": "ProcessCompletion",
					"Url": "https://sfdev.panasonicfa.com/if/connections/validate-process-completion-1/start"
					}
					]
					}
						{
						"ProcessCompletionWorkflowUrls": [
						{
						"Process": "ProcessCompletion",
						"Url": "https://sfdev.panasonicfa.com/if/connections/validate-process-completion-1/start"
						}
						]
						}

Flow Start Kafka Connection	smpwf-eh-0-dev.servicebus.windows.net:9093	smpwf-eh-0-dev.servicebus.windows.net:9093	smpwf-eh-0-dev.servicebus.windows.net:9093	smpwf-eh-0-dev.servicebus.windows.net:9093	smpwf-eh-0-dev.servicebus.windows.net:9093	S
Flow Start Kafka Topic	panasonic.workflow.runtime.trigger.topic	panasonic.workflow.runtime.trigger.topic	panasonic.workflow.runtime.trigger.topic	panasonic.workflow.runtime.trigger.topic	panasonic.workflow.runtime.trigger.topic	panasonic.workflow.runtime.trigger.topic

Flow Start Kafka Auth	saslplain	saslplain	saslplain	saslplain	saslplain	saslplain
Flow Start Kafka UserName	$ConnectionString	$ConnectionString	$ConnectionString	$ConnectionString	$ConnectionString	$ConnectionString
Flow Start Kafka PassWord	Endpoint=sb://smpwf-eh-0-dev.servicebus.windows.net/;SharedAccessKeyName=runtime_trigger;SharedAccessKey=FkkQpg4KUSQ6RixXj8fXCoDDIDx9TLccn+AEhFaHWgE=;EntityPath=panasonic.workflow.runtime.trigger.topic	Endpoint=sb://smpwf-eh-0-dev.servicebus.windows.net/;SharedAccessKeyName=runtime_trigger;SharedAccessKey=FkkQpg4KUSQ6RixXj8fXCoDDIDx9TLccn+AEhFaHWgE=;EntityPath=panasonic.workflow.runtime.trigger.topic	Endpoint=sb://smpwf-eh-0-dev.servicebus.windows.net/;SharedAccessKeyName=runtime_trigger;SharedAccessKey=FkkQpg4KUSQ6RixXj8fXCoDDIDx9TLccn+AEhFaHWgE=;EntityPath=panasonic.workflow.runtime.trigger.topic	Endpoint=sb://smpwf-eh-0-dev.servicebus.windows.net/;SharedAccessKeyName=runtime_trigger;SharedAccessKey=FkkQpg4KUSQ6RixXj8fXCoDDIDx9TLccn+AEhFaHWgE=;EntityPath=panasonic.workflow.runtime.trigger.topic	Endpoint=sb://smpwf-eh-0-dev.servicebus.windows.net/;SharedAccessKeyName=runtime_trigger;SharedAccessKey=FkkQpg4KUSQ6RixXj8fXCoDDIDx9TLccn+AEhFaHWgE=;EntityPath=panasonic.workflow.runtime.trigger.topic	Endpoint=sb://smpwf-eh-0-dev.servicebus.windows.net/;SharedAccessKeyName=runtime_trigger;SharedAccessKey=FkkQpg4KUSQ6RixXj8fXCoDDIDx9TLccn+AEhFaHWgE=;EntityPath=panasonic.workflow.runtime.trigger.topic
Flow End Kafka Connection	smpwf-eh-0-dev.servicebus.windows.net:9093	smpwf-eh-0-dev.servicebus.windows.net:9093	smpwf-eh-0-dev.servicebus.windows.net:9093	smpwf-eh-0-dev.servicebus.windows.net:9093	smpwf-eh-0-dev.servicebus.windows.net:9093	smpwf-eh-0-dev.servicebus.windows.net:9093

Flow End Kafka Topic	panasonic.workflow.runtime.waitforevent.topic	panasonic.workflow.runtime.waitforevent.topic	panasonic.workflow.runtime.waitforevent.topic	panasonic.workflow.runtime.waitforevent.topic	panasonic.workflow.runtime.waitforevent.topic	panasonic.workflow.runtime.waitforevent.topic
Flow End Kafka Auth	saslplain	saslplain	saslplain	saslplain	saslplain	saslplain
Flow End Kafka UserName	$ConnectionString	$ConnectionString	$ConnectionString	$ConnectionString	$ConnectionString	$ConnectionString
Flow End Kafka PassWord	Endpoint=sb://smpwf-eh-0-dev.servicebus.windows.net/;SharedAccessKeyName=runtime_waitforevent;SharedAccessKey=epq7vrydYiyB/x8J3nhrVSfsleWXdmPFJ+AEhNed6uU=;EntityPath=panasonic.workflow.runtime.waitforevent.topic	Endpoint=sb://smpwf-eh-0-dev.servicebus.windows.net/;SharedAccessKeyName=runtime_waitforevent;SharedAccessKey=epq7vrydYiyB/x8J3nhrVSfsleWXdmPFJ+AEhNed6uU=;EntityPath=panasonic.workflow.runtime.waitforevent.topic	Endpoint=sb://smpwf-eh-0-dev.servicebus.windows.net/;SharedAccessKeyName=runtime_waitforevent;SharedAccessKey=epq7vrydYiyB/x8J3nhrVSfsleWXdmPFJ+AEhNed6uU=;EntityPath=panasonic.workflow.runtime.waitforevent.topic	Endpoint=sb://smpwf-eh-0-dev.servicebus.windows.net/;SharedAccessKeyName=runtime_waitforevent;SharedAccessKey=epq7vrydYiyB/x8J3nhrVSfsleWXdmPFJ+AEhNed6uU=;EntityPath=panasonic.workflow.runtime.waitforevent.topic	Endpoint=sb://smpwf-eh-0-dev.servicebus.windows.net/;SharedAccessKeyName=runtime_waitforevent;SharedAccessKey=epq7vrydYiyB/x8J3nhrVSfsleWXdmPFJ+AEhNed6uU=;EntityPath=panasonic.workflow.runtime.waitforevent.topic	Endpoint=sb://smpwf-eh-0-dev.servicebus.windows.net/;SharedAccessKeyName=runtime_waitforevent;SharedAccessKey=epq7vrydYiyB/x8J3nhrVSfsleWXdmPFJ+AEhNed6uU=;EntityPath=panasonic.workflow.runtime.waitforevent.topic

Tenant ID	1	1	1	1	1	1
Coating Group Type Name	Cathode	Anode	Cathode	Anode	Cathode	Anode
Create Group URL	https://amcapidev.eastus.cloudapp.azure.com/inventoryruntime/carriers/receive	https://amcapidev.eastus.cloudapp.azure.com/inventoryruntime/carriers/receive	https://amcapidev.eastus.cloudapp.azure.com/inventoryruntime/carriers/receive	https://amcapidev.eastus.cloudapp.azure.com/inventoryruntime/carriers/receive	https://amcapidev.eastus.cloudapp.azure.com/inventoryruntime/carriers/receive	https://amcapidev.eastus.cloudapp.azure.com/inventoryruntime/carriers/receive
Authentication URL	https://amcapidev.eastus.cloudapp.azure.com/usermgmtauthentication/realms/Default/protocol/openid-connect/token	https://amcapidev.eastus.cloudapp.azure.com/usermgmtauthentication/realms/Default/protocol/openid-connect/token	https://amcapidev.eastus.cloudapp.azure.com/usermgmtauthentication/realms/Default/protocol/openid-connect/token	https://amcapidev.eastus.cloudapp.azure.com/usermgmtauthentication/realms/Default/protocol/openid-connect/token	https://amcapidev.eastus.cloudapp.azure.com/usermgmtauthentication/realms/Default/protocol/openid-connect/token	https://amcapidev.eastus.cloudapp.azure.com/usermgmtauthentication/realms/Default/protocol/openid-connect/token
devtest@panasonic.com	devtest@panasonic.com	devtest@panasonic.com	devtest@panasonic.com	devtest@panasonic.com	devtest@panasonic.com	devtest@panasonic.com
Authentication Password	xxx	xxx	xxx	xxx	xxx	xxx

Authentication Type	password	password	password	password	password	password
Authentication Client	account	account	account	account	account	account
Authentication Scope	openid profile email	openid profile email	openid profile email	openid profile email	openid profile email	openid profile email
ElasticSearch URL	http://elastic:kW1T74pcH5V715vP5Ad42Afs@sm-main-elasticsearch-es-http.sm-elasticsearch.svc.cluster.local:9200	http://elastic:kW1T74pcH5V715vP5Ad42Afs@sm-main-elasticsearch-es-http.sm-elasticsearch.svc.cluster.local:9200	http://elastic:kW1T74pcH5V715vP5Ad42Afs@sm-main-elasticsearch-es-http.sm-elasticsearch.svc.cluster.local:9200	http://elastic:kW1T74pcH5V715vP5Ad42Afs@sm-main-elasticsearch-es-http.sm-elasticsearch.svc.cluster.local:9200	http://elastic:kW1T74pcH5V715vP5Ad42Afs@sm-main-elasticsearch-es-http.sm-elasticsearch.svc.cluster.local:9200	http://elastic:kW1T74pcH5V715vP5Ad42Afs@sm-main-elasticsearch-es-http.sm-elasticsearch.svc.cluster.local:9200
Create Service	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
Create Ingress	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
Ingress	/if/connections/external/v1/coating/cathode	/if/connections/external/v1/coating/anode	/if/connections/external/v1/press	/if/connections/external/v1/pressing/anode	/if/connections/external/v1/sb/cathode	/if/connections/external/v1/sb/anode

	Path			ing/cathode			

List of Workflows

The following are the process workflows:

Workflow Name	Description
Mixing Workflow	Mixing is the first step, and this mixture is called slurry and is made up of active materials, conductive additives, solvents, and binders.
Consumption Mixing	Mixing consumes material (different parts for cathode and anode) and adjusts the quantity.
Mixing Manual QC	Manual QC inspection measures Viscosity, pH value, etc. to check its consistency.
Coating Workflow	Foil and Slurry (prepared in mixing process) are used as raw material and coil produced.
Pressing Workflow	Coated coil is used as input material and Pressed Coil is produced.
Sunburn Workflow	Pressed coil is used as Raw material and SB Coil produced.
Slitting Workflow	SB coil slit to 8 pancakes
Unstable Rewind	The Pancake that is produced in Slitting, further goes through unstable slitting (Equipment will send for unevenness). Some pancakes go through Unstable slit process to make a proper pancake
Normal Rewindinding Workflow	Anode pancake rewinding-ensures that the components are ready for winding. Some pancakes (Anode even pancake) alone go through the Rewinding process.
ARP Workflow	All pancakes go through the ARP process, before moving to the Winding (Assembly) process. It creates/gets CASSETTE in ARP and automates the release of tested and approved battery cells.
ARPDrop Workflow	Indicates that a battery cell has failed the ARP and has been rejected due to quality or safety issues.
DefectStatusAvailability	It checks if any process has issues it will report before further making pancake as available.
Optimized Winding Workflow	Pancake and other Raw material are used as input material and Jelly Roll is produced.
Optimized Assembly Can Workflow	Starts with Can loading where Jelly roll is mounted on Can.
Assembly Consumption Workflow	Raw materials are consumed.

Assembly NG Consumption Workflow	NG CANs raw materials are consumed.
AssemblyTrayWorkflow	Create Assembly Lot -> Tray -> Add CAN, Duplicate ID Scan handling, Record Inspection (CAN), Data Collection (CAN). AL logical group is created.
Optimized Assembly TrayWorkflow	Create Assembly Lot -> Tray -> Add CAN, Duplicate ID Scan handling, Record Inspection (CAN), Data Collection (CAN). AL logical group is created.
Formation Workflow Type-E	This WF is started for contracts FC042 to FC046. The steps may not be executed in sequence. Initial charging/discharging process to activate and balance battery cells. We set counts for Cans in Assembly Lot.
Formation Workflow Type-C	Any cells that do not pass the required specifications are extracted from the assembly process. We unlink NG cells from tray.
Formation Workflow Type-D	Cells that fail testing or quality control checks are marked for rejection.
Formation Workflow Type-A	This WF is started when the first formation contract is received for Cell tray. Cell Id and other attributes are added in the CANs.
Optimized VI Can Based	CAN are used as input material and CELL ID is produced followed by Welding, inspection. The process ends with Boxing, where CANs are loaded to Cell trays.
VI Tray Based	CAN are used as input material and CELL ID is produced followed by Welding, inspection. The process ends with Boxing, where CANs are loaded to Cell trays.
Packaing Box Based Workflow	Packaging process packs the produced cells in the labelled boxes (Inner & Outer) and move the boxes inside the pallet.
Packing Pallet Based Workflow	Once the pallet has the required cells available after packaging box process, it will be ready to shift to the customer.
CellCheckMissingProcessDataWorkflow_Assembly	This WF sets the respective evaluation code and mapped status using reason code and Station group fetch from master data.
CellCheckMissingProcessDataWorkflow_Formation	
CellCheckMissingProcessDataWorkflow_VI	This WF sets the respective evaluation code and mapped status using reason code and Station group fetch from master data.
CellCheckSameDischargeFactor	This WF sets the respective evaluation code and mapped status using reason code and station name fetch from master data. It sets the evaluation

	code only if max retries are exceeded form the defined retry limit and this limit is fetched by station name and defect code in the master data.
CellCheckSameEquipmentDischarged	This WF sets the respective evaluation code and mapped status using reason code and station name fetch from master data. It sets the evaluation code only if max retries are exceeded form the defined retry limit and this limit is fetched by station name and defect code (here defect code passed from Adapter Setting i.e. "OVRLIMIT") in the master data.
CellCheckPreventionOfMixingOfDifferentProductsWorkFlow	
CellCheckTimeElapseWorkFlow	
CellCheckTimeOnLine24hrWorkFlow	
CellCheck_TJ_Judgement	This particular cell check is done to do a final decision on what happens to Cells marked as TJ At specific check points.
CellCheckAssemblyMissingProcessDataLogic	
CellCheckLotAverageWeight_VI	
CellCheckLotAvergaeCapacity_Formation	
GhostCell_Formation	This checks the ghost cells and sets some attribute for formation.
GhostCell_Packaging	This checks the ghost cells and sets some attribute for packaging.
GhostCell_VI	
NGExistingRate_Assembly	This workflow raise an alert and create a ticket if the number of Defect cells in the assembly Lot is higher than a certain threshold percentage.

The following are the common workflows:

Workflow Name	Description
Optimized Aging Workflow	Monitoring system used to track the aging process of each process during their lifecycle.
DefectRollup Workflow	Monitors defects which are in open, review or processed state.
Timer Expiration Workflow	
ManualQC Workflow	Sample tickets are created for processes.
Process Completion Workflow	Inventories are set to Available on fulfilling certain steps.

Propagation Workflow	Link defect to lot
MaterialExhaustTimerDeactivation Workflow	
Cell Check Trigger Workflow	This is the main cell check which is called from the cell check adaptor and it further calls the other cell checks based on the configuration.
Pitch Handling	On the process end, data for pitches within the coil are provided which is recorded in MES-C (Smart hive platform).
DailyQC	Daily Inspection ticket creation
EvaluationCodeWorkflow	Based on priority eval/defect code is allotted.

List of Grafana Alerts

Grafana alerts feature allows you to monitor metrics and send notifications when defined conditions are met. They are primarily used to ensure the health, performance, and reliability of systems, applications, and infrastructure.

The Grafana alerts listed below are used for monitoring and alerting purposes:

Category	Alert	Description	Alert Level (Warning/Error/Low/Medium/Critical)
Hardware	NodeCPUHighUsage	It gives an alert when CPU usage > threshold value for each node (>80%)	Medium
	NodeMemoryHighUsage	It gives an alert when Memory usage > threshold value for each node (>80%)	Medium
	NodeHDDHighUsage	It gives an alert when Disk Space usage > threshold value for each node (>80%)	Medium
	HighHighNumberOfVMSnapshots	High Number of Snapshots	Low
Infrastructure	PostgreSQLDown	postgresql instance is down	High
	PostgresqlRestarted	Postgresql restarted	Medium
	KafkaConsumerLag	Kafka consumer has an increased lag (setup some threshold time e.g. 30 min)	High
	K8SNodeNotReady	Kubernetes node has been unready for a long time	High
	K8SNodeDiskPressure	Kubernetes node has disk pressure condition	High
	K8SPVCPending	Kubernetes PersistentVolumeClaim specified is pending	Low
	K8SPVError	Persistent volume {{ $labels.persistentvolume }} is in bad state	High
	K8SHPAScaleInability	Kubernetes Horizontal Pod Autoscaler(HPA) is unable to scale	Medium
	K8SHPAMetricsUnavailability	Kubernetes Horizontal Pod Autoscaler(HPA) unable to	Medium

		collect metrics	
	K8SHPAScaleMaximum	Kubernetes Horizontal Pod Autoscaler(HPA) has hit maximum number of desired pods	Low
	K8SHPAUnderUtilized	Kubernetes Horizontal Pod Autoscaler(HPA) underutilized	Low
	K8SClientCertificateExpiring	Kubernetes client certificate expires next week	Low
	CoreDNSPanicCount	CoreDNS Panic Count is high. A high panic count indicates that CoreDNS is repeatedly crashing, which can disrupt DNS resolution in the cluster.	High
	PVCleanupAlert	Alert for cleaning up unused PVs	
	SSLCertificateExpiry	SSL certificate expiry (< 7 days) (instance {{ $labels.instance }})	High
PVC	PVC-Unused	Alert for cleaning up using script for PVCs currently in Bound state and Unused by any pod.	Medium
	PVcleanup	Alert for cleaning up using script for PVs in Released state	Medium
Service/Pods/Jobs	K8SJobFailed	A Kubernetes job has failed i.e. Job {{ $labels.namespace }}/{{ $labels.job_name }} failed to complete	
	K8SPodCrashLooping	Pod {{ $labels.namespace }}/{{ $labels.pod }} is crash looping Pod is crash looping 'n' no. of times/5minutes	
	K8SPodNotHealthy	Pod {{ $labels.namespace }}/{{ $labels.pod }} has been in a non-running state for longer than 15 minutes.	
	K8SPodNotHealthy	Pod ($namespace/$pod) has been in a non-running state	Medium

		for longer than 30 minutes.	
	K8SJobFailed	Job {job_name} failed to complete	Low
Micro Service	microservice_errors	It gives an alert when {job="panasonic-workorder-coreapp-api-monitoring"} >0.	Critical
Container	ContainerKilled	Container killed (instance {{ $labels.instance }})	Low
	ContainerAbsent	Container absent (instance {{ $labels.instance }})	Low
	ContainerHighCPUUtilization	Container High CPU utilization (instance {{ $labels.instance }})	Medium
	ContainerHighMemoryUsage	Container Memory usage is above 80% Container High Memory usage (instance {{ $labels.instance }})	Medium
	ContainerHighVolumeUsage	Container Volume usage is above 80% Container Volume usage (instance {{ $labels.instance }})	Low
	cri-dockerd-events-alert	It monitors the health of critical system components, the CRI Docker daemon (cri-dockerd), and its associated events. It identifies unhealthy system or Kubernetes-related events that indicate operational issues.	Critical
Network	BlackBoxHttpFailure	HTTP status code is not 200-399 Blackbox probe HTTP failure (instance {{ $labels.instance }})	Medium
	BlackBoxHttpSlow	Blackbox probe slow HTTP (instance {{ $labels.instance }})	Medium
	HostUnusualNetworkThroughputIn	Host unusual network throughput in (instance {{ $labels.instance }})	Medium

	HostUnusualNetworkThroughputOut	Host unusual network throughput out (instance {{ $labels.instance }})	Medium
Nginx	NginxHighHttp4xxErrorRate	Too many HTTP requests with status 4xx (> 5%)	Medium
	NginxHighHttp5xxErrorRate	Too many HTTP requests with status 5xx (> 5%)	High