Srivastava, SudeepX

[Company name]  [Company address]

Project Deployment and Scan Status

# Objective([DCG2-1791](https://jira.devtools.intel.com/browse/DCG2-1791) Create appropriate Project):- Publish and consume Deployment/Scan/Pod status through Kafka.

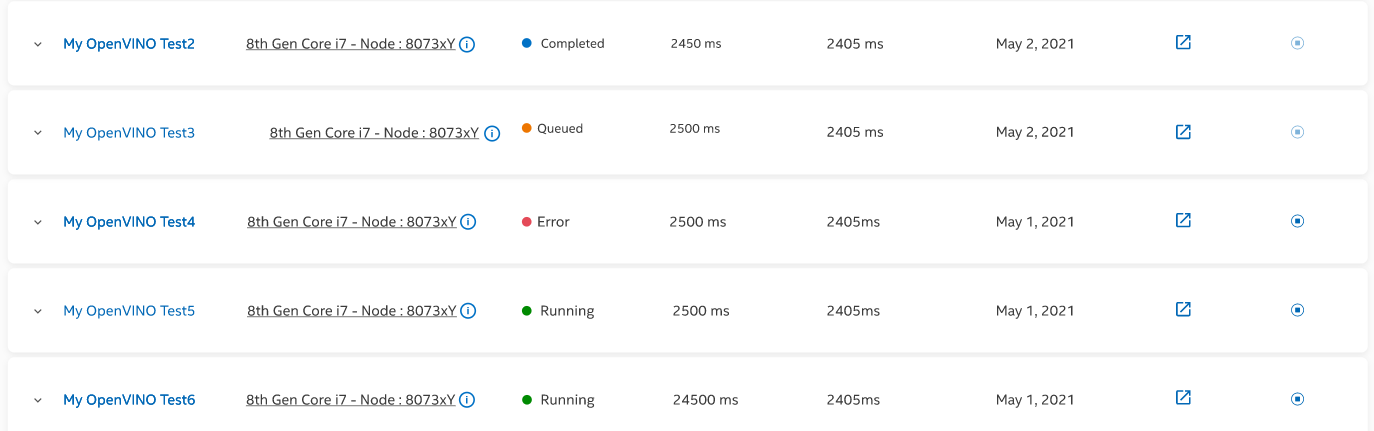
# Solution Approach: - We suggest event based modeling through Kafka to post and retrieve Deployment and Scan status and capture it in DB (Gen2\_Deployment table and Gen2\_PodStatusTable) through BYOC Consumer.

Graphical user interface, application

Description automatically generated

As could be depicted from above diagram, every service which is initiating Deployment Or Scan will push a message in new Kafka Topic(containing only monitoring statuses). **It will be a Separate topic that will hold only messages related to Resource statuses**

# Dashboard UX Screenshot for status display:-



# 2-level monitoring and Payload Structure: - There would be 2 level of monitoring status captured

* **Deployment Level status**
  + Deployment level will contain status related to a deployment
    - Queued status
    - Scan status
  + Deployment level status will be stored in gen2\_deployment table and would be primarily driven based on “ComparisonID” and “DeploymentID”.
  + **“ComparisonID”(random UUID ):-** corresponds to ID which uniquely identifies every deployment.
  + **“DeploymentID**”(random **UUID)**:- corresponds to ID which uniquely identifies every/list of containers which are deployed as part of “ComparisonID”.
  + Ex. In case of multi container/Helm/Docker Compose where multiple containers are deployed corresponding to a single deployment, there would be multiple “DeplymentID’s mapped with single “ComparisonID”.
* **Container or Pod Level status**
  + Container level status would be driven from “Gen2\_Pod\_Status” table
  + This status will be pushed using existing event based mechanism from openshift-int-ms.
* For capturing running status of pods “ComparisionID” and “DeploymentID” should be stored as pod labels before sending to kafka topic,
* “ComparisionID” and “DeploymentID” also should be persisted in the DB table that every ms primarily uses to refer deployment related data in order to trace back deployment details that may be useful during re-launch functionality, where ComparisionID/DeploymentID’s could be send back to the origin service to relaunch the same helm chart or docker compose file etc.

## Payload Structure:-

|  |
| --- |
| Deployment Status Payload: |

|  |
| --- |
| Pod/Container status message Payload:- |

## **Byoc consumer**:- Currently we are using Byoc-ms as consumer, going forward we can create separate Monitoring consumer service. Byoc consumer will start listening over incoming message and would consume all messages using custom Header “messageType” and would retrieve comparisionId/DeploymentId and update “Gen2\_Deployment” table with Latest status.

## **Byoc Get API to show Target and Pod status on Dashboard**: will fetch “Status” column value from Gen2\_Deployment table that will be used to display project Target status.

## **Producer MS services**: - will generate random UXID value corresponding to ComparisionID and DeploymentID and push status to Kafka topic.

## **Refactor existing Pod monitoring service** as well to send status using the same topic instead of directly calling Byoc-ms through HTTP call

# DashBoard API payload:

|  |  |
| --- | --- |
| URL:- https://byoc-devcloud-r2-dev.apps.cfa.devcloud.intel.com/api/v1/pod/ | |
| GET API Response |  |
| {  "userId": 22,  "pageNumber": null,  "pageSize": null,  "projectCount": 1,  "listProjects": [  {  "projectDTO": {  "projectId": 5699,  "name": "demoprojectfirst",  "description": "firstproject",  "createTime": "2021-07-21T14:04:27.058+00:00",  "updateTime": "2021-07-21T14:07:30.757+00:00",  "containerTerminationTime": null,  "assignedContainers": [  5700,  5706,  5703  ]  },  "targets": [  {  "comparisonId": "e1e768d0-c1a2-42a2-9491-84f7b5dadb73",  "targetName": "core-i5 gen8",  "targetStatus": "ADDED",  "executionTime": 69,  "lastJobExecution": 518949,  "createTime": "2021-07-27T14:20:36.761",  "deployments": [  {  "deploymentId": "8a068d9a7ac94007017ac9685e0f000f",  "userId": 22,  "containerId": 5703,  "tags": null,  "containerCreationDate": "2021-07-21T14:06:44.189+00:00",  "compute": "cpu",  "cpu": "core",  "gen": "gen8",  "variant": "i5",  "status": "ERROR",  "podIp": "10.135.38.11 ",  "podId": "523ed6dc-d1eb-4c69-897f-c65e51469e9d",  "podName": "2-secondcontainer-58d7d7d474-lc8vw",  "imageName": "docker.io/tarun142258/private:latest",  "executionTime": 0,  "containerName": "secondcontainer",  "path": null,  "lastJobExecution": 518949,  "createTime": "2021-07-23T05:42:25.156",  "manuallyStopped": false  },  {  "deploymentId": "8a068d9a7ac94007017ac9685e0c000e",  "userId": 22,  "containerId": 5700,  "tags": null,  "containerCreationDate": "2021-07-21T14:05:02.866+00:00",  "compute": "cpu",  "cpu": "core",  "gen": "gen8",  "variant": "i5",  "status": "ADDED",  "podIp": "10.135.38.10 ",  "podId": "4242c62a-ff08-4462-84ec-9f753cd5c5cd",  "podName": "1-firstdemocontainer-c877bc49c-bj5tp",  "imageName": "docker.io/rupak91/openshift:script",  "executionTime": 69,  "containerName": "firstdemocontainer",  "path": null,  "lastJobExecution": 518949,  "createTime": "2021-07-27T14:20:36.761",  "manuallyStopped": false  },  {  "deploymentId": "8a068d9a7ac94007017ac9685e07000d",  "userId": 22,  "containerId": 5706,  "tags": null,  "containerCreationDate": "2021-07-21T14:07:30.732+00:00",  "compute": "cpu",  "cpu": "core",  "gen": "gen8",  "variant": "i5",  "status": "ADDED",  "podIp": "10.135.38.9 ",  "podId": "08b3405e-e6c4-44ca-913d-45de6add23ff",  "podName": "0-thirdcontainer-5c8d59c464-sp7gc",  "imageName": "docker.io/rupak91/openshift:script",  "executionTime": 69,  "containerName": "thirdcontainer",  "path": null,  "lastJobExecution": 518949,  "createTime": "2021-07-27T14:20:37.784",  "manuallyStopped": false  }  ],  "manuallyStopped": false  }  ]  }  ]  } | |

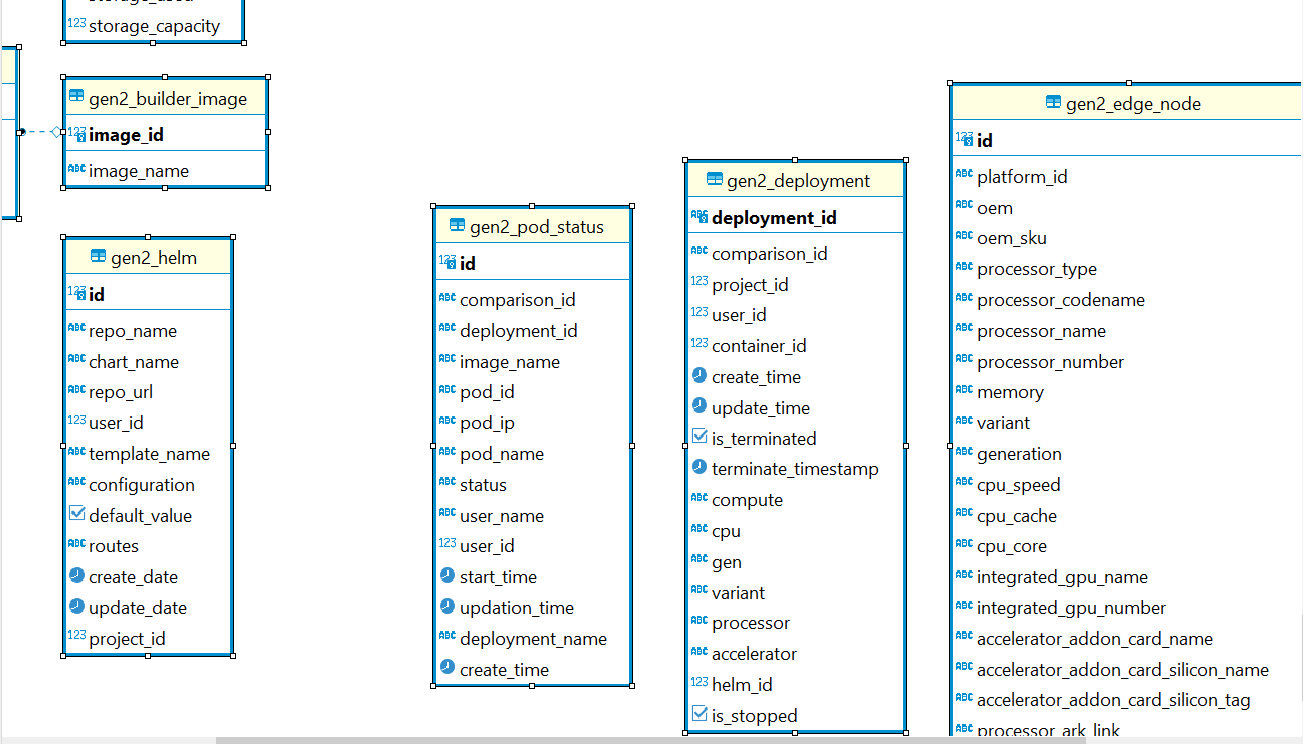
# Database changes:-

**Gen2\_Deployment\_Table modification:** Currently this table contains multiple ID’s level information we will update it to contain corresponding value level info for every ID like ProjectID/ContainerId etc.

**Graphical user interface, application

Description automatically generated**

# ER Diagram:-



# Assumptions:-

## Every Service will have to create a producer and push status to Kafka topic.

* Separate topic to capture deployment status
* RE-launch - Respective resources will have to go through the payload and comment if additional values need to be fetched as part of GET API to perform re-deploy