Elastic GPU Service (EGS): API Specification

Effective resource management of GPUs across Kubernetes Clusters

Introduction

This document provides EGS REST API (/api/v1) details to manage the Slice workspace and GPU resources programmatically.

Rapid increase in the development of fine-tuned SLM/LLM models-based GenAI Apps has created a scarcity of specialized GPUs like A100/H100 (or new B100/B200 resources) across the providers/regions, resulting in a need for efficient resource and schedule management of these GPU nodes in one or more clusters to maximize the availability of the GPUs to the Users/Teams/Pipelines and improve the utilization of GPUs.

EGS documents

This document provides EGS API details.

- For Admin guide please see the documentation on website
- For User guide please see the documentation on website
- For Installation guide please see the documentation on github repo

Table of contents

Overview

EGS backend exposes a number of APIs to create and manage Slice workspace and GPU provision request (GPRs) resources in one or more EGS managed clusters. EGS manages cluster GPU resources and dynamically allocates GPU nodes (GPUs) to slice workspaces based on the request priority and other request attributes.

EGS API can be used by:

- LLM-Ops/ML-Ops/RAG pipelines from outside the cluster to create and manage slice workspaces and GPRs for those workspaces
- LLM/ML/RAG control plane or ML/LLM-Ops in cluster services to create and manage slice workspaces and GPRs for those workspaces

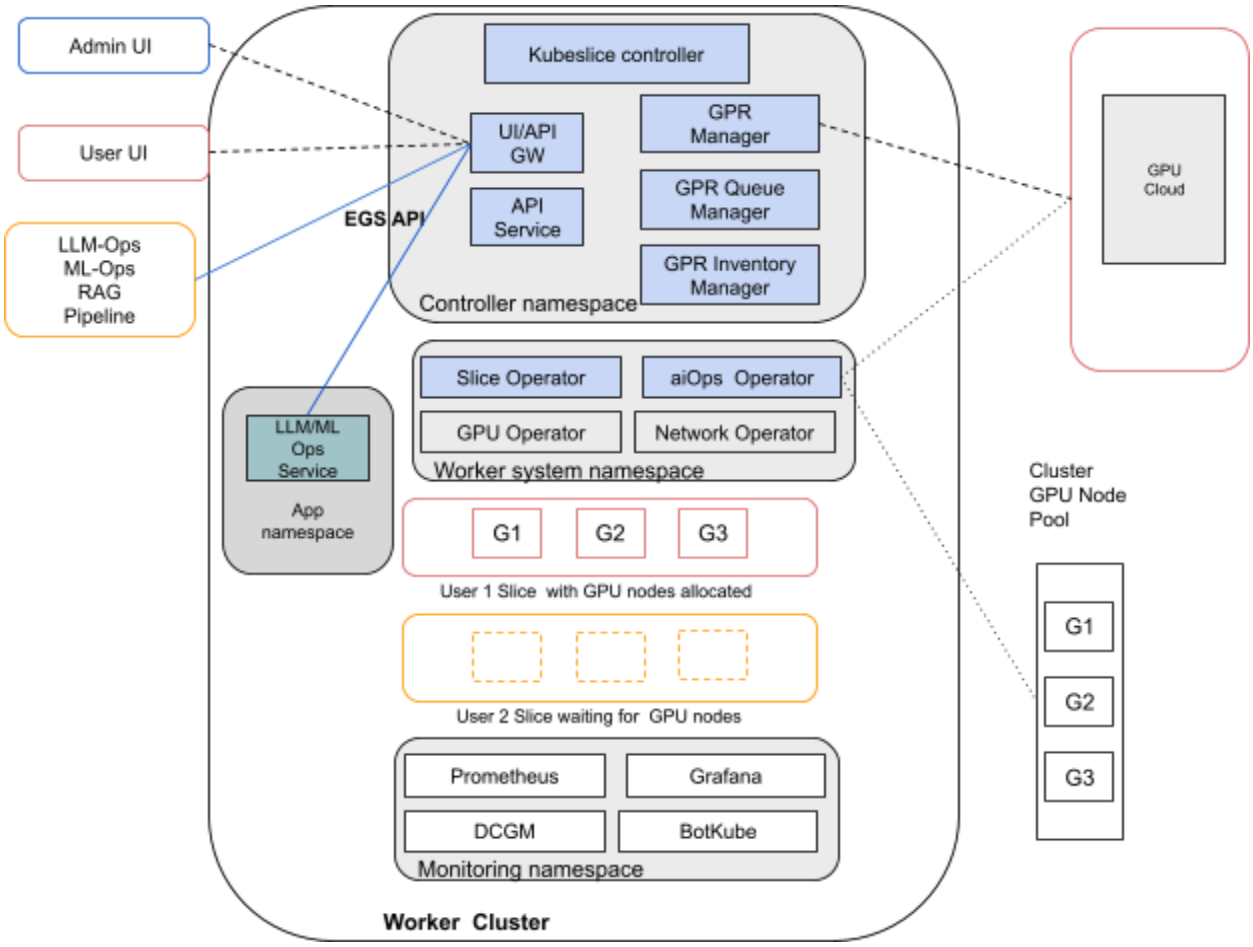

Fig 1: EGS API single cluster reference model

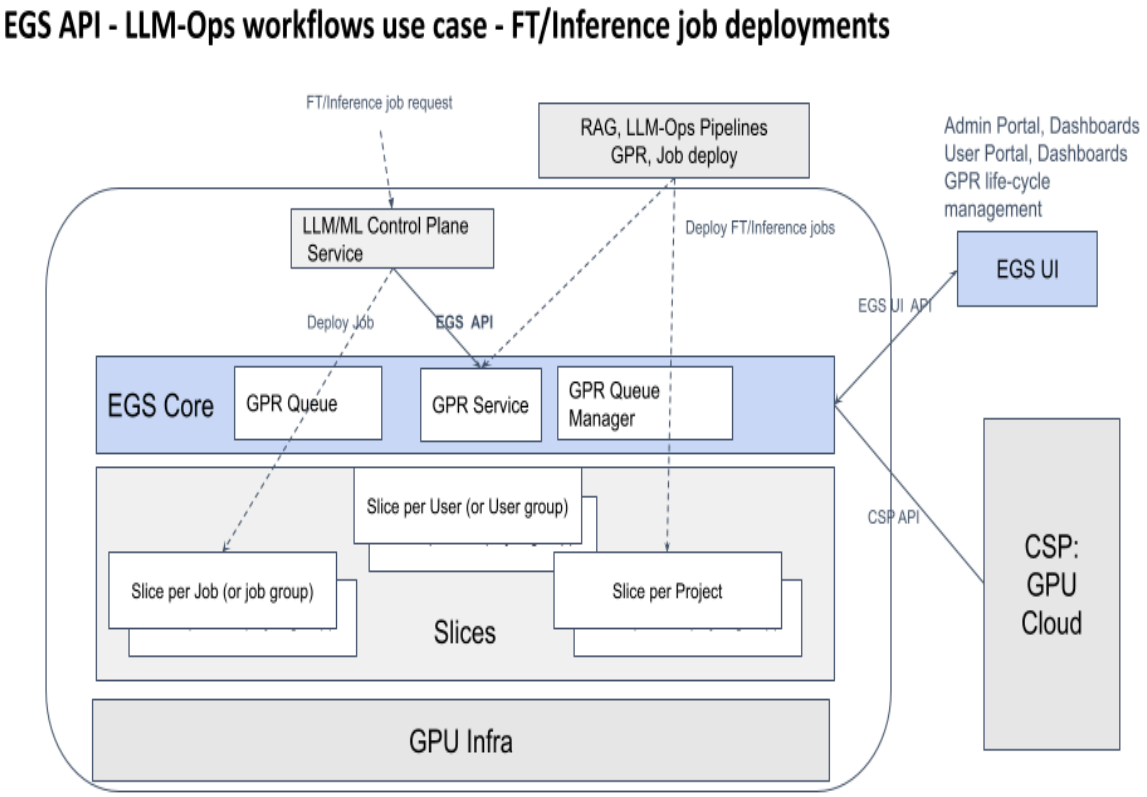

Fig 2: Example EGS API use case - LLM-Ops workflows

Install EGS and Access Admin Portal

EGS provides a combination of helm charts, CLI and shell scripts for easy installation procedure.

Please follow the installation steps specified in EGS installation documentation

Once installation is complete, please see Access the Admin Portal step to access the Portal to create API Tokens.

API Token for Authentication

An API token will be used to authenticate to EGS to obtain a bearer token for other APIs.

There are different roles associated with API Tokens.

1. Owner role (LLM-Ops/ML-Ops admin) API Token
    a. This token will allow the client to create and manage:
        i. User Slice workspace related resources and
            1. Slice, namespaces, User, etc.
        ii. GPRs (GPU provision requests)
        iii. View other read only resources - like Inventory, etc.
2. Viewer role API Token
    a. This token will allow the client to GET operations on Slice workspaces, Inventory, GPRs etc. resources.

Generating an API Token

API tokens can be generating with the following steps:

1. Login in to EGS Portal (UI) as Admin (or User)
2. Select “API Tokens” in left panel
3. Click “+Add API Token”
    a. Pop up menu to add the token
        i. Name, description, pull down menu for “Owner”
        ii. Create API Token
4. Copy the “API Token”

Manage API Tokens

Admin (User) can create and delete the API Tokens as needed using API Tokens page workflows.

Using EGS APIs

EGS provides the following APIs to create and manage Slice workspaces and GPRs (GPU provision requests) associated with Slice Workspaces.

Note: You can create a Slice workspace by following the steps in the Admin guide, if needed.

Note: You have to use the **Owner** role token to create Slice workspace programmatically.

API POSTMAN Collection

The EGS API POSTMAN collection is available at:
https://github.com/kubeslice-ent/egs-installation/tree/main/egs-apis

EGS APIs

Elastic GPU Service (EGS) backend exposes the following APIs to create and manage Slice workspaces, get inventory details and create and manage GPU provision requests (GPRs) for the Slice workspaces.

Authentication

Before using any other APIs Client needs to authenticate using the API Token to get a bearer token for the API calls.

Request

```
Unset
POST /api/v1/auth
{
    "apiKey": "apiKey"
}
```

- apiKey: <API token>

Note: This API does not need Authorization: bearer token

Responses

```
Unset
{
    "statusCode": 200,
    "status": "OK",
    "message": "Success",
    "data": {
        "token": ""
    }
}
```

- data.token: <token>
    - Use this token for other API calls in the Authorization header
        - Authorization bearer token

Response: Success

```
Unset
{
    "statusCode": 200,
    "status": "OK",
    "message": "Success",
    "data": {
        "token":
"eyJhbGciOiJSUzI1NiIsImtpZCI6IlNTYWZkejFDOE12VnZmZi1TeU9iUkZBVUtQcjhLcjdZaVFkSl
RUd1BZM0UifQ.eyJhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG
9jYWwiXSwiZXhwIjoxNzI5NDU0ODg5LCJpYXQiOjE3Mjk0NTM5ODksImlzcyI6Imh0dHBzOi8va3ViZ
XJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZXRlcy5pbyI6eyJuYW1lc3Bh
Y2UiOiJkZWZhdWx0Iiwic2VydmljZWFjY291bnQiOnsibmFtZSI6Imt1YmVzbGljZS1yYmFjLXJvLXN
saWNlLWJsdWUiLCJ1aWQiOiIyMDdmMzllOC0yMjUwLTQ3NTUtOGY2MC01ZWM1YzZjNzM3YjIifX0sIm
5iZiI6MTcyOTQ1Mzk4OSwic3ViIjoic3lzdGVtOnNlcnZpY2VhY2NvdW50OmRlZmF1bHQ6a3ViZXNsa
WNlLXJiYWMtcm8tc2xpY2UtYmx1ZSJ9.p_hZ3PSbXufy7d8wAdyLGn_hMjahRP0SVw6qDlCeP_FcPrI
HHH1hdWWiNXfSY-RggrdYATGj4iTrDcFgp52pY1pW4yywdGc9zWd19fyreT5UkoM8HvEkV5zbyNuf2t
QZRK6zbMjVS-HzwutSdE_M0tpx8zqO2uosLr29kkxG-3qe7DhoOsg04x0NjN9jNw3PQ9l6u48nSg0aU
wW2CJAK-7SAp4tUjMnXyh4wxop7mZwimls6VLUoCCqtyACpc9WZNxqO82nCKmnUf6sMTens37PSMz_e
lM0kMMMUmZhlAcbxy3-fGbHDfjOv1_iaF5K3mbqw5Id0k2-xXcTM7TkUBw"
    }
}
```

Response: ApiKey Secret Malformed

```
Unset
{
    "statusCode": 400,
    "status": "UNHANDLED_REQUEST_ERROR",
    "message": "Api Key Configuration is invalid. Request a new API-key or
contact administrator",
    "error": {
        "errorKey": "UNHANDLED_REQUEST_ERROR",
        "message": "Api Key Configuration is invalid. Request a new API-key or
contact administrator",
        "data": null
    }
}
```

Response: ApiKey Expired

```
Unset
{
    "statusCode": 401,
    "status": "UNAUTHENTICATED",
    "message": "Api Key has expired",
    "error": {
        "errorKey": "UNAUTHENTICATED",
        "message": "Api Key has expired",
        "data": null
    }
}
```

Response: Incorrect Key

```
Unset
{
    "statusCode": 401,
    "status": "UNAUTHENTICATED",
    "message": "Unauthenticated",
    "error": {
        "errorKey": "UNAUTHENTICATED",
        "message": "Unauthenticated",
        "data": null
    }
}
```

Example

```
Unset
curl --location --globoff '{{host}}/api/v1/auth' \
--data '{
    "apiKey": "apiKey"
}'
```

Using the Bearer Token in a Client

You use the /api/v1/auth generated token as part of the `Authorization` header in API requests.

Here's an example using `curl`:

```
Unset
curl -H "Authorization: Bearer <token>"
http://localhost:8080/api/v1/slice-workspace
```

Create Slice Workspace

Create a Slice workspace. Slice workspace is associated with one or more namespaces and a User.
You can create a slice workspace for a :

- training/fine tuning/Inference or any other ML Job
- Job group
- ML project
- User (or User group) to deploy, develop applications in the namespaces

Once you have a Slice workspace created you can request GPUs for the workspace using Create GPR (GPU provision requests) to get GPUs allocated for the workspace.

Once you have created a slice workspace use Generate KubeConfig for slice workspace API to get the KubeConfig file to access the kubernetes namespaces to deploy the AI workloads on the workspace namespaces.

Request

```
Unset
POST /api/v1/slice-workspace
{
    "workspaceName": "workspaceName",
    "clusters": [
        "cluster-name registered with EGS"
    ],
    "namespaces": [
```

```
        "namespace-name"
    ],
    "username": "John Doe",
    "email": "john.doe@avesha.io"
}
```

- **workspaceName**: Name of the workspace name, can be User/Team/Job-workspace/Project name
- **cluster-name**: cluster's name registered with EGS.
- **namespace-name**: cluster namespace name to be associated with the workspace
- **username**: User (team/job/project) name associated with the workspace.

Responses

Response: Success

```
Unset
{
    "statusCode": 200,
    "status": "OK",
    "message": "Success",
    "data": {
        "workspaceName": "workspace-name"
    }
}
```

Response: Already Exists

```
Unset
{
    "statusCode": 409,
    "status": "Error",
    "message": "Already Exists",
    "data": null,
    "error": {
        "errorKey": "",
        "message": "Already Exists",
```

```
        "data": "{\"error\":{\"Message\":\"Error registering
Slice\",\"Status\":409,\"DetailedError\":{\"Errormessage\":\"sliceconfigs.contr
oller.kubeslice.io \\\"evening\\\" already
exists\",\"Reason\":\"AlreadyExists\",\"Details\":\"\"},\"StatusCode\":409,\"Re
ferenceDocumentation\":\"https://kubeslice.io/documentation/open-source/0.2.0/g
etting-started-with-cloud-clusters/installing-kubeslice/creating-a-slice\"}}"
    }
}
```

Response: Bad Request

Unset

```
{
    "statusCode": 422,
    "status": "Error",
    "message": "Bad Request",
    "data": null,
    "error": {
        "errorKey": "",
        "message": "Bad Request",
        "data": "{\"error\":{\"Message\":\"Error registering
Slice\",\"Status\":422,\"DetailedError\":{\"Errormessage\":\"admission webhook
\\\"vsliceconfig.kb.io\\\" denied the request:
SliceConfig.controller.kubeslice.io \\\"evening\\\" is invalid:
Spec.NamespaceIsolationProfile.ApplicationNamespaces.Clusters: Required value:
clusters\",\"Reason\":\"Invalid\",\"Details\":[{\"reason\":\"FieldValueRequired
\",\"message\":\"Required value:
clusters\",\"field\":\"Spec.NamespaceIsolationProfile.ApplicationNamespaces.Clu
sters\"}]},\"StatusCode\":422,\"ReferenceDocumentation\":\"https://kubeslice.io
/documentation/open-source/0.2.0/getting-started-with-cloud-clusters/installing
-kubeslice/creating-a-slice\"}}"
    }
}
```

Example

Unset

```
curl --location --globoff '{{host}}/api/v1/slice-workspace' \
```

```
--data-raw '{
    "workspaceName": "workspaceName",
    "clusters": [
        "worker-1"
    ],
    "namespaces" : [
        "namespace-1",
        "namespace-2"
    ],
    "username": "John Doe",
    "email": "john.doe@avesha.io"
}'
```

Delete Slice Workspace

This API can be used to delete a slice workspace. The request is queued and processed by the EGS backend. Use LIST workspaces API to see the workspaces and workspace details.

Request

Unset

```
DELETE /api/v1/slice-workspace
{
    "workspaceName": "workspace-name"
}
```

Response

Response: Success

Unset

```
{
    "statusCode": 200,
    "status": "OK",
    "message": "Success",
    "data": {}
```

```
}
```

List Slice Workspaces

List slice workspace API returns list of slice workspaces and workspaces details.

Request

```
Unset
GET /api/v1/slice-workspace/list
```

Response

```
Unset
{
    "statusCode": 200,
    "status": "OK",
    "message": "Success",
    "data": {
        "workspaces": [
            {
                "name": "workspace1",
                "overlayNetworkDeploymentMode": "no-network",
                "maxClusters": 16,
                "sliceDescription": "Created via slice workspace API",
                "clusters": [
                    "worker-1"
                ],
                "namespaces": [
                    {
                        "namespace": "namespace-3",
                        "clusters": [
                            "worker-1"
                        ]
                    }
                ]
```

```
            },
            {
                "name": "workspace2",
                "overlayNetworkDeploymentMode": "no-network",
                "maxClusters": 16,
                "sliceDescription": "Created via slice workspace API",
                "clusters": [
                    "worker-1"
                ],
                "namespaces": [
                    {
                        "namespace": "namespace-1",
                        "clusters": [
                            "worker-1"
                        ]
                    },
                    {
                        "namespace": "namespace-2",
                        "clusters": [
                            "worker-1"
                        ]
                    }
                ]
            }
        ]
    }
}
```

Example

Unset

```
curl --location --globoff '{{host}}/api/v1/slice-workspace/list'
```

Generate KubeConfig For Slice Workspace

Use this API to get the KubeConfig to access the Slice workspace namespaces associated with the cluster. You can use this KubeConfig to deploy the AI workloads in the namespaces.

Request

Unset

```
GET /api/v1/slice-workspace/kube-config
{
    "workspaceName": "workspace-name"
}
```

Response

Unset

```
{
    "statusCode": 200,
    "status": "OK",
    "message": "Success",
    "data": {
        "kubeConfig": "apiVersion: v1\nkind: Config\nclusters:\n    - name:
worker-1\n      cluster:\n        server: https://x.x.x.x\n
certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0ttCk1JSUVMVENDQXBXZ0F3SUJBZ0lSQUptUWNiTjdRUXZzYn
ExR3I0MFdFNsdUFKSDF2UE40agpic3VES3BKSThqNnUyWjFBS28yemRIdTBVTjRjclFhNWs0UmxaYm1
5djM3Vm83RytEdGZxUEpwVGNjQVFTYkFBCm4xTExwODNYUU5jUmFPem1RSlRCWHdwbWQyUXZ2ODVWb2
dvWFNGNGhCQXZCVXB4T2NVTkRycnlyUlk1VU9NUTEKb1p5RGMrRnNVaTNsak4rV2M5NmNWM2s9Ci0tL
S0tRU5EIENFUlRJRklDQVRFLS0tLS0K\nusers:\n    - name: afternoon-workspace\n
user:\n        token:
eyJhbGciOiJSUzI1NiIsImtpZCI6ImtsMUNpZDR5ZmlKb2dsVzVOTjFmdGN0emhKbjR2dEh5RWN2RmY
taTRNS28ifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9
zZXJ2aWNlYWNjb3VudC9I6qvdkapR8_YTlugVdhdxIhrCppWhKsCeX0g3wT2tmjV90rcJXia1jejUmh
fIwkxOj2R2-N57UfSB5dIb1DPZ_XX21yxJhcBua3GAe0v908wiMRTj_BlLVZyOj31d2d_0M4aQl2JCe
0tUg4pL0BBEavk-cUXLHC8wOlT2BlmMqq6G1m3ZZPcrW4Eljt17OZVpGlMIjR4ZN1ag78fSFhr2trPk
QNEYygIGJBI5oLDYH04SHURvJMA-LIqWodoLJq-s7uhgZFi4UfIp5m6bw2UATWt8AyffDnLww\ncont
exts:\n    - name: workspace1@worker-1\n      context:\n        cluster:
worker-1\n        user: workspace1\n        namespace:
namespace-3\ncurrent-context: afternoon-workspace@worker-1\n"
    }
}
```

Example

```
Unset
curl --location --globoff '{{host}}/api/v1/slice-workspace/kube-config' \
--data '{
    "workspaceName": "workspaceName"
}'
```

List Inventory

List GPU nodes available for allocation requests (GPRs) in the cluster. These nodes are managed and allocated by the EGS.

Request

```
Unset
GET /api/v1/inventory/list
```

Response

```
Unset
{
    "statusCode": 200,
    "status": "OK",
    "message": "Success",
    "data": {
        "items": [
            {
                "gpuNodeName":
"gke-dev-2-gke-cluste-dev-2-gke-cluste-9d71107a-hs19",
                "gpuShape": "n1-standard-8",
                "gpuModelName": "Tesla-T4",
                "instanceType": "n1-standard-8",
                "clusterName": "worker-1",
                "memory": 15360,
                "gpuCount": 1,
                "gpuTempThreshold": "",
                "gpuPowerThreshold": "",
```

```
                "cloudProvider": "gcp",
                "region": "",
                "nodeHealth": "Healthy",
                "gpuNodeStatus": "NODE_AVAILABLE"
            }
        ]
    }
}
```

Example

```
Unset
curl --location 'http://x.x.x.x:8080/api/v1/inventory/list'
```

List Inventory by Slice Workspace

List GPU nodes currently allocated to the Slice Workspace.
  Note: Only one GPR is provisioned on a Slice Workspace at any given time. Other GPRs requested will be queued and allocated one at a time based on the priority.

Request

```
Unset
GET /api/v1/inventory?sliceName=workspace1
```

Responses

Response: Success

Unset


```
{
    "statusCode": 200,
    "status": "OK",
    "message": "Success",
    "data": {
        "items": [
            {
                "nodeName": [
                    "gke-dev-2-gke-cluste-dev-2-gke-cluste-9d71107a-hs19"
                ],
                "instanceType": "n1-standard-8",
                "memory": 15360,
                "totalGpus": 1,
                "gpuShape": "Tesla-T4"
            }
        ]
    }
}
```


Response: Not Found

Unset


```
{
    "statusCode": 404,
    "status": "Error",
    "message": "Not Found",
    "data": null,
    "error": {
        "errorKey": "",
        "message": "Not Found",
        "data": "{\"error\":{\"Message\":\"Error occurred while fetching the
inventory
details\",\"Status\":404,\"DetailedError\":{\"Errormessage\":\"sliceconfigs.con
troller.kubeslice.io \\\"workspace2\\\" not
found\",\"Reason\":\"NotFound\",\"Details\":\"\"},\"StatusCode\":404,\"Referenc
eDocumentation\":\"NA\"}}"
```

```
    }
}
```

Example

```
Unset
curl --location 'http://localhost:8080/api/v1/inventory?sliceName=evening'
```

Create GPR (GPU provision request)

Create GPU provision requests for a slice workspace. A GPR is a GPU allocation request to the EGS to provision GPUs to workspace for a requested duration.

GPU provision request includes:

- GPU node instance type
  - You can get instance types from List Inventory API call
- GPU type
  - You can get GPU types from List Inventory API call
- Duration
  - Duration of the allocation
- Number of Nodes and number of GPUs
- Priority

Note: You can create many GPRs, only one GPR (high priority) GPR will be provisioned at any given time. Other GPRs will be in queued state. Once the high priority GPR is complete the next highest priority GPR for the workspace will be provisioned.

Request

```
Unset
POST /api/v1/gpr
{
    "gprName":"gpr1",
    "sliceName":"job-workspace1",
    "clusterName":"worker-1",
    "numberOfGPUs":1,
    "instanceType":"n1-standard-8",
```

```
    "exitDuration":"0d0h5m",
    "numberOfGPUNodes": 1,
    "priority":201,
    "memoryPerGPU":15,
    "gpuShape":"Tesla-T4"
}
```

- gprName: provision request name
- instanceType: GPU node instance type (from the List Inventory details)
- numberOfGPUNodes: number of GPU nodes requested
- Priority: priority of the request
    - Low: 00-100
    - Med: 100-200
    - High: 200-300
- gpuShape: name of the GPU type (from the List Inventory details)
- exitDuration: GPU allocation duration - format: xxdxxhxxm

Responses

Response: Success

```
Unset
{
    "statusCode": 200,
    "status": "OK",
    "message": "Success",
    "data": {
        "gprId": "gpr-1bb22a0b-9388"
    }
}
```

- gprId: GPR identifier
    - Note down the Id - need it for other GPR APIs

Response: Bad Request

```
Unset
{
```

```
    "statusCode": 422,
    "status": "Error",
    "message": "Bad Request",
    "data": null,
    "error": {
        "errorKey": "",
        "message": "",
        "data": "{\"error\":{\"Message\":\"Error while fetching GPR wait
time\",\"Status\":422,\"DetailedError\":{\"Errormessage\":\"\\\"exitDuration\\\
" is not allowed to be
empty\",\"Reason\":\"\",\"Details\":\"\"},\"StatusCode\":422,\"ReferenceDocumen
tation\":\"NA\"}}"
    }
}
```


Example

Unset

```
curl --location --globoff '{{host}}/api/v1/gpr' \
--data '{
    "gprName":"gr1",
    "sliceName":"workspaceName",
    "clusterName":"worker-1",
    "numberOfGPUs":1,
    "instanceType":"n1-standard-8",
    "exitDuration":"0d0h5m",
    "numberOfGPUNodes": 1,
    "priority":201,
    "memoryPerGPU":15,
    "gpuShape":"Tesla-T4"
}'
```

Delete GPR

Delete a queued GPR. You can delete not provisioned GPR using this API call.

Note: If the GPR is already provisioned (GPUs allocated to the workspace) then use GPR Update API call with Early Release option to release the GPUs and delete the GPR.

Request

```
Unset
DELETE /api/v1/gpr
{
    "gprId": "gpr-e4e38fc5-4742"
}
```

Response

```
Unset
{
    "statusCode": 200,
    "status": "OK",
    "message": "Success",
    "data": {}
}
```

Response: Cannot delete

GPR provision was complete - completed GPRs cannot be deleted.

```
Unset
{
    "statusCode": 500,
    "status": "Error",
    "message": "Cannot delete",
    "data": null,
    "error": {
        "errorKey": "",
        "message": "UNKNOWN",
```

```
        "data": "{\"error\":{\"Message\":\"Error while deleting
GPR\",\"Status\":500,\"DetailedError\":{\"Errormessage\":\"Cannot delete GPR in
Successful
state\",\"Reason\":\"\",\"Details\":\"\"},\"StatusCode\":500,\"ReferenceDocumen
tation\":\"NA\"}}"
    }
}
```

Example

```
Unset
```

Update GPR

Use this API to update GPR details or early release a provisioned GPR.

Note: If the GPR is already provisioned (GPUs allocated to the workspace) then use GPR Update API call with Early Release option to release the GPUs and delete the GPR. If GPR is still in the queued state then use Delete GPR API to delete the GPR.

Request

```
Unset

PUT /api/v1/gpr
{
    "gprId":"gpr-e4e38fc5-4742",
    "priority": 101,
    "earlyRelease": false,
    "gprName": "gpr1"
}
```

- gprId: GPR Id from GPR create response
- priority: new priority value
    - Low: 00-100
    - Med: 100-200
    - High: 200-300

Responses

Response: Success

```
Unset
{
    "statusCode": 200,
    "status": "OK",
    "message": "Success",
    "data": {}
}
```

Response: Unable Early Release

Unable to Early Release as GPR is not provisioned yet - still in the queue. List GPRs by workspace will return all the GPRs for the Slice workspace.

```
Unset
{
    "statusCode": 500,
    "status": "Error",
    "message": "Unable to Early Release",
    "data": null,
    "error": {
        "errorKey": "",
        "message": "Unable to Early Release",
        "data": "{\"error\":{\"Message\":\"Error while updating a
GPR\",\"Status\":500,\"DetailedError\":{\"Errormessage\":\"Cannot update early
release of GPR in Queued
state\",\"Reason\":\"\",\"Details\":\"\"},\"StatusCode\":500,\"ReferenceDocumen
tation\":\"NA\"}}"
    }
}
```

Example

```
Unset
curl --location --globoff --request PUT '{{host}}/api/v1/gpr' \
--data '{
    "gprId":"gprId",
    "priority": 101
}'
```

List GPRs by Slice Workspace

List GPRs associated with a slice workspace. The response includes GPR status and other details for each GPR associated with the workspace.

Request

```
Unset
GET /api/v1/gpr/list?sliceName=workspace1
```

Response: Success

List the GPRs associated with the Slice workspace.

```
Unset
{
    "statusCode": 200,
    "status": "OK",
    "message": "Success",
    "data": {
        "items": [
            {
                "gprId": "gpr-1bb22a0b-9388",
                "sliceName": "workspace1",
                "clusterName": "worker-1",
                "numberOfGPUs": 1,
                "numberOfGPUNodes": 1,
                "instanceType": "n1-standard-8",
```

```
            "memoryPerGPU": 15,
            "priority": 201,
            "gpuSharingMode": "Virtual Machine",
            "estimatedStartTime": "2024-10-22T11:12:38Z",
            "estimatedWaitTime": "0h0m0s",
            "exitDuration": "1h15m0s",
            "earlyRelease": true,
            "gprName": "gpr1",
            "gpuShape": "Tesla-T4",
            "multiNode": false,
            "dedicatedNodes": false,
            "enableRDMA": false,
            "enableSecondaryNetwork": false,
            "status": {
                "provisioningStatus": "Queued",
                "failureReason": "",
                "numGpusAllocated": 0,
                "startTimestamp": "",
                "completionTimestamp": "",
                "cost": "",
                "nodes": [
                    "gke-dev-2-gke-cluste-dev-2-gke-cluste-9d71107a-hs19"
                ],
                "internalState": "Queued",
                "retryCount": 0,
                "delayedCount": 0
            }
        },
        {
            "gprId": "gpr-5d278fd3-e868",
            "sliceName": "workspace1",
            "clusterName": "worker-1",
            "numberOfGPUs": 1,
            "numberOfGPUNodes": 1,
            "instanceType": "n1-standard-8",
            "memoryPerGPU": 15,
            "priority": 101,
            "gpuSharingMode": "Virtual Machine",
            "estimatedStartTime": "2024-10-22T06:30:39Z",
            "estimatedWaitTime": "0h0m0s",
            "exitDuration": "0h10m0s",
            "earlyRelease": false,
            "gprName": "rag-gpr-1",
            "gpuShape": "Tesla-T4",
```

```
            "multiNode": false,
            "dedicatedNodes": false,
            "enableRDMA": false,
            "enableSecondaryNetwork": false,
            "status": {
                "provisioningStatus": "Complete",
                "failureReason": "",
                "numGpusAllocated": 0,
                "startTimestamp": "",
                "completionTimestamp": "2024-10-22T06:41:17Z",
                "cost": "",
                "nodes": [
                    "gke-dev-2-gke-cluste-dev-2-gke-cluste-9d71107a-hs19"
                ],
                "internalState": "Released",
                "retryCount": 0,
                "delayedCount": 0
            }
        },
        {
            "gprId": "gpr-784039ab-0cda",
            "sliceName": "workspace1",
            "clusterName": "worker-1",
            "numberOfGPUs": 1,
            "numberOfGPUNodes": 1,
            "instanceType": "n1-standard-8",
            "memoryPerGPU": 15,
            "priority": 101,
            "gpuSharingMode": "Virtual Machine",
            "estimatedStartTime": "2024-10-22T07:41:17Z",
            "estimatedWaitTime": "0h2m40s",
            "exitDuration": "0h10m0s",
            "earlyRelease": false,
            "gprName": "ft-gpr-1",
            "gpuShape": "Tesla-T4",
            "multiNode": false,
            "dedicatedNodes": false,
            "enableRDMA": false,
            "enableSecondaryNetwork": false,
            "status": {
                "provisioningStatus": "Provisioned",
                "failureReason": "",
                "numGpusAllocated": 1,
                "startTimestamp": "",
```

```
                    "completionTimestamp": "2024-10-22T07:52:17Z",
                    "cost": "",
                    "nodes": [
                        "gke-dev-2-gke-cluste-dev-2-gke-cluste-9d71107a-hs19"
                    ],
                    "internalState": "Provisioned",
                    "retryCount": 0,
                    "delayedCount": 0
                }
            }
           }
        ]
    }
}
```

- status.provisionStatus: Queued, Provisioned, Complete, Error

Response: Not Found

Unset

```
{
    "statusCode": 404,
    "status": "Error",
    "message": "Slice Workspace Not Found",
    "data": null,
    "error": {
        "errorKey": "",
        "message": "Slice Workspace Not Found",
        "data": "{\"error\":{\"Message\":\"Error in listing GPR's for slice
evening-x\",\"Status\":404,\"DetailedError\":{\"Errormessage\":\"sliceconfigs.c
ontroller.kubeslice.io \\\"evening-x\\\" not
found\",\"Reason\":\"NotFound\",\"Details\":\"\"},\"StatusCode\":404,\"Referenc
eDocumentation\":\"NA\"}}"
    }
}
```

Example

```
Unset
curl --location --globoff '{{host}}/api/v1/gpr/list?sliceName=workspace1'
```