Migration from Azure to Google Cloud for Project Sunbird: (Final Stage of Deployment)

As part of the ongoing migration of Project Sunbird from Azure to Google Cloud Platform (GCP), the infrastructure has been successfully provisioned on GCP using Infrastructure as Code (IaC) through Terraform. This includes the creation of networking resources such as Virtual Private Cloud (VPC) and subnets, as well as other essential components like Load Balancers, Google Kubernetes Engine (GKE) clusters, node pools, and associated resources.

Current Status

- **Infrastructure Setup**: Completed on GCP using Terraform.
- **Networking**: VPC, subnets, and Load Balancer configured through Terraform.
- **GKE and Node Pools**: Both GKE clusters and node pools are fully provisioned.
- **Application Deployment**: Applications have been deployed on GKE.

Issue Identified

Upon deployment, several application pods are failing to start or function as expected. After thorough investigation, it has been determined that the application code still retains dependencies on Azure services, which causes issues when running in the GCP environment. These dependencies need to be refactored to point to equivalent GCP services.

Next Steps for Resolution

To ensure successful application deployment on GCP, the following actions are required from the development team:

1. **Code Refactoring for Cloud-Specific Services**: The current code is hardcoded to use Azure-specific services These services need to be replaced with their GCP counterparts, such as Google Cloud Storage, Google Secret Manager, and other relevant GCP services.

2. **Configuration Files Needing Changes**: Below are some key files that need to be updated, **although there are other areas that may require similar changes:**

```
if [ "$CLOUD_SERVICE" == "azure" ]; then
    # Use az cli to upload the file
    az storage blob upload --account-name ${AZURE_STORAGE_ACCOUNT} \
                           --account-key ${AZURE_KEY} \
                           --container-name ${AZURE_CONTAINER} \
                           --name fulldb-$backup_date.sql.bz2 \
                           --type block \
                           --file "$backup_dir/fulldb-$backup_date.sql.bz2"
fi
```

```yaml
play_http: &play_http
  secret_key: "secretKey"
schema: &schema
  base_path: "https://{{ .Values.global.azure_storage_account_name }}.blob.core.windows.net/{{ .Values.global.azure_public_container_name }}/schemas/local"
graph_passport: &graph_passport
  key_base: "secretKey"
installation: &installation
  id: ekstep
channel: &channel
  default: "in.ekstep"
objectcategorydefinition: &objectcategorydefinition
  keyspace: "{{ .Values.global.env }}_category_store"

global:
  redis: *redis
  elasticsearch: *es
  neo4j: *neo4j
  cassandra: *cassandra
  zookeeper: *zookeeper
  kafka: *kafka
  play_http: *play_http
  schema: *schema
  graph_passport: *graph_passport
  installation: *installation
  channel: *channel
  objectcategorydefinition: *objectcategorydefinition
  provisioningAnnotations: *provisioningAnnotations
```

```
    at org.apache.flink.runtime.executiongraph.DefaultExecutionGraphBuilder.buildGraph(DefaultExecutionGraphBuilder.j
311) ~[flink-dist_2.12-1.13.5.jar:1.13.5]
    at org.apache.flink.runtime.scheduler.DefaultExecutionGraphFactory.createAndRestoreExecutionGraph(DefaultExecutio
phFactory.java:107) ~[flink-dist_2.12-1.13.5.jar:1.13.5]
    at org.apache.flink.runtime.scheduler.SchedulerBase.createAndRestoreExecutionGraph(SchedulerBase.java:342) ~[flin
st_2.12-1.13.5.jar:1.13.5]
    at org.apache.flink.runtime.scheduler.SchedulerBase.<init>(SchedulerBase.java:190) ~[flink-dist_2.12-1.13.5.jar:1
5]
    at org.apache.flink.runtime.scheduler.DefaultScheduler.<init>(DefaultScheduler.java:122) ~[flink-dist_2.12-1.13.5
:1.13.5]
    at org.apache.flink.runtime.scheduler.DefaultSchedulerFactory.createInstance(DefaultSchedulerFactory.java:132) ~[
k-dist_2.12-1.13.5.jar:1.13.5]
    at org.apache.flink.runtime.jobmaster.DefaultSlotPoolServiceSchedulerFactory.createScheduler(DefaultSlotPoolServi
hedulerFactory.java:110) ~[flink-dist_2.12-1.13.5.jar:1.13.5]
    at org.apache.flink.runtime.jobmaster.JobMaster.createScheduler(JobMaster.java:340) ~[flink-dist_2.12-1.13.5.jar:
.5]
    at org.apache.flink.runtime.jobmaster.JobMaster.<init>(JobMaster.java:317) ~[flink-dist_2.12-1.13.5.jar:1.13.5]
    at org.apache.flink.runtime.jobmaster.factories.DefaultJobMasterServiceFactory.internalCreateJobMasterService(Def
JobMasterServiceFactory.java:107) ~[flink-dist_2.12-1.13.5.jar:1.13.5]
    at org.apache.flink.runtime.jobmaster.factories.DefaultJobMasterServiceFactory.lambda$createJobMasterService$0(De
tJobMasterServiceFactory.java:95) ~[flink-dist_2.12-1.13.5.jar:1.13.5]
    at org.apache.flink.util.function.FunctionUtils.lambda$uncheckedSupplier$4(FunctionUtils.java:112) ~[flink-dist_2
1.13.5.jar:1.13.5]
    at java.util.concurrent.CompletableFuture$AsyncSupply.run(Unknown Source) ~[?:?]
    at java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) ~[?:?]
    at java.util.concurrent.FutureTask.run(Unknown Source) ~[?:?]
    at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(Unknown Source) ~[?:?]
    at java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) ~[?:?]
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) ~[?:?]
    at java.lang.Thread.run(Unknown Source) ~[?:?]
ed by: org.apache.flink.fs.shaded.hadoop3.org.apache.hadoop.fs.azure.AzureException: java.lang.IllegalArgumentExcepti
URI 'wasbs://@.blob.core.windows.net/checkpoint/questionset-publish' has a malformed WASB authority, expected contain
ame.Authority takes the form wasb://[<container name>@]<account name>
    at org.apache.flink.fs.shaded.hadoop3.org.apache.hadoop.fs.azure.AzureNativeFileSystemStore.createAzureStorageSes
(AzureNativeFileSystemStore.java:1086) ~[flink-azure-fs-hadoop-1.13.5.jar:1.13.5]
    at org.apache.flink.fs.shaded.hadoop3.org.apache.hadoop.fs.azure.AzureNativeFileSystemStore.initialize(AzureNativ
```

```
neo4j: &neo4j
  enabled: true
  host: "neo4j"
  port: 7687
  boltPort: 8687
  shard_id: 1
  image:
    repository: sunbirded.azurecr.io/neo4j
    tag: "3.3.0"
  commonAnnotations:
    "helm.sh/hook-weight": "-5"
  provisioning:
    annotations: *provisioningAnnotations

global:
  redis: *redis
  elasticsearch: *es
  cassandra: *cassandra
  zookeeper: *zookeeper
  kafka: *kafka
  postgresql: *postgresql
  druid: *druid
  neo4j: *neo4j
  provisioningAnnotations: *provisioningAnnotations
  keycloak_kid_keys_configmap: keycloak-kids-keys
  keycloak_key_configmap: keycloak-key
  kong_desktop_device_consumer_names_for_opa: '["portal", "deskto

migration:
  cassandra:
    waitTime: 120
    jarUrl: "https://github.com/user/repo/releases/download/v1.0/
  elasticsearch:
```

Testing and Validation:

- Once the code dependencies on Azure are fully removed and refactored for GCP, thorough testing needs to be conducted to ensure compatibility and functionality within the GCP environment.

- The DevOps team is ready to assist in redeploying the application once the code modifications are complete.