Apache Beam GSoC 2025 Proposal

Enrique Job Calderón Olalde - Mexico City

Implementing IaC and cloud automated features for Apache Beam

Issue

https://issues.apache.org/jira/browse/GSOC-273

Simplify management of Beam infrastructure, access control and permissions via Platform features

Motivation

Apache Beam is an open source project that I have been trying to use to implement in some of my projects where I require parallel data processing for big quantities of data.

I am more used to infrastructure and sysadmin work. While searching for issues to solve in order to be part of GSoC I noticed this issue where they require this knowledge so I decided I could help through solving this problem, following best practices and a clear solution.

Proposal

For the first problem I have been working with @pabloem in https://github.com/apache/beam/pull/34141 to create a python system that could be used that stores the GCP resources with their day of creation, this to make it clearer which resources are older and should be deleted to avoid unnecessary resource usage.

For the second issue I would like to follow GCP best practices making a central IaC deployment so that when running tests the resources are managed in a single place, enforcing strict access policies. Accomplishing this would help detect any breakage because we would have a history of how resources worked before. We know database storage is expensive but we could use other kinds of storage to implement our own cloud logging.

Benefits

Implementing IaC in the project would help with the resource usage making the test's bill smaller and clearer. This is because we would detect any resource leak and old unused resources in an easier way. Algo improving security and scalability thanks to following GCP the best practices.

I know Apache Beam is a Open Source project and even though it is sponsored by google we do not need to waste money.

Deliverables

The milestones of this project are:

- Automated cleaning tool for GCP test environments
    - Continue developing the automated cleaning tool implementing an audit trail so that we can also get what, why and when resources are created. The tools is being developed in this issue.

      In the technical side, this is being implemented using python and GCP APIs, making use of a storage bucket to avoid using any database. The idea is that it would run on schedule, every hour. The cleaner detects and stores the state of the resources running on GCP.

      The code for the base class is something like:

```
class Cleaner:
        def __init__(self, type, project_id):
            self.type = ""
            self.project_id = project_id

        def list_instances():
        # List all the instances that exist right now

        def delete_instance():
        # Delete an specific instance

        def clean_old_instances():
        # Compare instances based on:
        # - time
        # - usage
        # - name
        # According to them select the ones ready to be deleted
```

      This would exist for each resource type with a main script calling each resource type cleaner. The cleaner follows the logic:

      1. Get the resources that exist now, log it.
      2. Get the resources that existed when the script was last run.
      3. Create a list of resources that should be deleted but still exist.

4. Delete the resources on a filter based on a regular expression over the resource name, resource type, time alive and confirming they are not being used.
5. Log and notify the responsible.

- Implement IaC for all Beam Infrastructure
    - Privilege management with Terraform

      Adding modules implementing roles to interact and acquire privileges for Beam resources. This will help to create an audit trail and upgrade from the current process where you need to contact the administrators and then they manually grant privileges.

      For this we would need a terraform project over the main apache/beam repository, inside the infra directory. The main module would need all permissions and each submodule just the necessary permissions, implementing the roles based on the practices created on the security part. Admin, Commiter, Community, Bot/CICD. Any change on the infrastructure would produce an alert to the administrator.

      Terraform is an IaC solution that focuses on managing infrastructure and can work with GCP so we can make it cleaner looking than using the APIs directly

    - Stateless Infra management with terraform

      Things like virtual machines and kubernetes clusters will be managed stateless with terraform. Helping with the inventory, privileges and controlling costs of resources. Some advantages of stateless is that we can avoid deleting misconfigured automated runs.

      The idea is adding Terraform because its one of the best ways of implementing stateless infra, being a big project for managing infrastructure based on the stateless architecture.

      Here is a example of IaC on terraform:

```
variable "project" {}
variable "region" {}
variable "zone" {}

provider "google" {
  project = var.project
  region  = var.region
}
```

```
resource "google_compute_instance" "default" {
  name         = "example-instance"
  machine_type = "n1-standard-1"
  zone         = var.zone

  boot_disk {
        initialize_params {
        image = "debian-cloud/debian-11"
        }
  }

  network_interface {
        network = "default"
        access_config {}
  }

  service_account {
        email  = "default"
        scopes = ["https://www.googleapis.com/auth/cloud-platform"]
  }

  metadata_startup_script = <<-EOT
        #!/bin/bash
        echo "Instance created on $(date)"
        # Now we load the script to allow ASF infra to take over the host
  EOT
}
```

- Create key management solutions with best practices
    - Creating an utility for users to create service account keys and store them in a secure way.

      Using the GCP standard key management service we can automate the rotation of keys and keep the audit of each generated key. The proposal is to use the GCP APIs to keep track of the keys and their age and when a new service account key is needed we can track which user is responsible for it and the reason for the key.

      The way of accomplishing this is implementing a python program to create, store and retrieve keys, this way the user would need to provide certain information to create a key. Things like: name, description, accounts related, access, administrator. This would keep an audit log of who, how, when and why, limiting the permissions that only the managers know about. And we could implement the

key rotation best practices inside the module, something like rotate all the keys. With the key retrieval we would cover the best practice of not hardcoding keys.

The code for the endpoints would look like:

```
def create_key():
        # Create the key based on:
        # Name
        # Description
        # Accounts
        # Access permissions
        # Alert user

def rotate_key():
        # Based on time
        # Generate and recreate the related keys

def retrieve_key():
        # Get a key
        # Used only by automated scripts

def delete_key():
        # When a key is not being used, allow us to delete it
```

The endpoint would only be accessed by administrators and the required users.

- Automated security monitoring and policy violation detection
    - Ingest audit logs and create alerts

      Ingesting the audit logs and analysing them we can classify them in search of suspicious ones, triggering an alert. This can go to the audit log of GCP or even to the OpenTelemetry collector. This can be done with python.

      The monitoring/logging tool follows the logic:
        1. An application generates logs
        2. The logs are ingested to GCP, if native logs are implemented greate, else implement logs to GCP using terraform, mainly for the audit logs.(This could be also done with OpenTelemetry.
        3. Configure automatic alerts inside GCP logging for any unusual log.

Time commitments

The Google Summer of Code starts officially on June 2 and ends on September 1.

Dates	Number of weeks	Description	Hours per week for GSoC
Now - May 24	7	Regular university time. I will look at the documentation and code needed to continue with the development of the cleaner.	10 - 15
May 26 - August 08	16	Summer time. Work on the code for the project, make tests, work side by side with the mentor.	35-40
August 11 - August 29	3	Return to university. Finish the missing code, last tests and presentation of the code.	10 - 15

About Me

I am Enrique Calderón. I am a Computer Engineering student at UNAM (National Autonomous University of Mexico). I've been part of the Free and Open Source (FOSS) movement and freedom of knowledge community. .My CV

I have been working in a Laboratory (LIDSoL) at my university where I am in charge of the infrastructure and DevOps. Some of the things we have done is hosting some public linux distributions mirrors, a tor node and some internal services using IaC tools. This has given me the knowledge related to the IaC and system management allowing me to interact with logging tools and more.

I would love to participate in this Google Summer of Code to contribute with my knowledge and time to make the internet a better place.