Here's a **complete explanation** of your Terraform configuration to help you confidently present this to your client.

**🧾 Overview**

This Terraform code sets up a **Google Cloud Monitoring and Alerting infrastructure** for the project **hsbc-12609073-peakplat-dev**. It includes:

* **Log-based metrics** for Kubernetes, Flink, and XDS logs
* **Resource utilization alerts** for VMs, Kubernetes nodes, and Cloud SQL
* **Notification channels** via **xMatters Webhook** and **Email**
* **Alert documentation** using templatefile() for each alert
* **Secrets management** via Google Secret Manager

**☁️ 1. Provider Configuration**

hcl

CopyEdit

provider "google" {

project = var.project\_id

region = var.region

}

* Connects Terraform to GCP using project ID and region variables.

**🔐 2. Secret Manager Integration**

hcl

CopyEdit

data "google\_secret\_manager\_secret\_version" ...

* Securely retrieves **xMatters credentials** (username/password) from **Secret Manager** for use in webhook authentication.

**📢 3. Notification Channels**

**a. xMatters Webhook**

hcl

CopyEdit

resource "google\_monitoring\_notification\_channel" "xmatters\_webhook" ...

* Sends alerts to xMatters using **Basic Auth** and a URL (var.xmatters\_webhook\_url).

**b. Email Notification**

hcl

CopyEdit

resource "google\_monitoring\_notification\_channel" "email" ...

* Sends alert emails to the address specified in var.notification\_email.

**📊 4. Log-Based Metric: Kube Error Logs**

hcl

CopyEdit

resource "google\_logging\_metric" "log\_based\_metrics\_dev" ...

* Creates a log-based metric tracking ERROR logs in the kube-system namespace.

**🚨 5. Alert Policy: Kube Error Logs**

hcl

CopyEdit

resource "google\_monitoring\_alert\_policy" "Log\_Based\_Metrics\_DEV" ...

* Triggers an alert when error logs are detected based on the above log metric.
* Uses:
  + **Condition** on logging.googleapis.com/user/Kube\_Error\_Logs
  + **Threshold**: log count > 0 over 15 minutes
  + **Documentation**: from kube\_error\_logs\_doc.tpl

**🖥️ 6. Alert Policy: VM & K8s Resource Utilization**

hcl

CopyEdit

resource "google\_monitoring\_alert\_policy" "vm\_resource\_utilization\_alert" ...

Includes 3 conditions:

**a. VM Memory > 80%**

* Uses agent.googleapis.com/memory/percent\_used
* 15 min window, threshold: 80%

**b. K8s Node Memory > 50%**

* Uses kubernetes.io/node/memory/allocatable\_utilization
* Threshold: 50%

**c. K8s Node CPU > 50%**

* Uses kubernetes.io/node/cpu/allocatable\_utilization
* Threshold: 50%

**🧠 7. Alert Policy: VM High CPU**

hcl

CopyEdit

resource "google\_monitoring\_alert\_policy" "vm\_high\_cpu\_utilization" ...

* Triggers when agent.googleapis.com/cpu/utilization > 80%
* Uses a **15-minute** window and **mean** alignment

**🧮 8. Alert Policy: Cloud SQL Memory Utilization**

hcl

CopyEdit

resource "google\_monitoring\_alert\_policy" "cloud\_sql\_utilization" ...

* Alerts when Cloud SQL memory usage > 70%
* Uses metric: cloudsql.googleapis.com/database/memory/utilization
* Region filter: europe-west2

**🧮 9. Alert Policy: Cloud SQL CPU Utilization**

hcl

CopyEdit

resource "google\_monitoring\_alert\_policy" "cloud\_sql\_cpu\_utilization" ...

* Triggers when Cloud SQL CPU > 70%
* Uses metric: cloudsql.googleapis.com/database/cpu/utilization

**🔧 10. Log-Based Metric: Flink Logs**

hcl

CopyEdit

resource "google\_logging\_metric" "flink\_log\_alert\_metric" ...

* Captures ERROR logs from Flink pods in Kubernetes.

**🚨 11. Alert Policy: Flink Error Logs**

hcl

CopyEdit

resource "google\_monitoring\_alert\_policy" "flink\_log\_alert\_policy" ...

* Uses the Flink log metric
* Trigger if any error appears within the 10-minute alignment window

**📡 12. Log-Based Metric: XDS Logs**

hcl

CopyEdit

resource "google\_logging\_metric" "xds\_error\_logs\_alerts" ...

* Captures specific log payloads related to XDS issues like RD10014, RD10035, etc.

**🚨 13. Alert Policy: XDS Error Logs**

hcl

CopyEdit

resource "google\_monitoring\_alert\_policy" "xds\_error\_logs\_alerts" ...

* Fires if any of the predefined XDS error messages occur
* Uses a **60-second** alignment period for rapid detection

**📄 14. Documentation with templatefile()**

Each alert uses:

hcl

CopyEdit

documentation {

content = templatefile("${path.module}/alert\_docs/<template>.tpl", {

project\_id = var.project\_id

})

mime\_type = "text/markdown"

}

* Dynamically loads alert documentation content from .tpl templates.
* You can include project-specific variables like ${project\_id} for customization.

**📬 15. Notification Flow**

Each alert sends messages to:

* **xMatters Webhook**: for incident response escalation
* **Email**: for team visibility or logging

**🧼 16. Best Practices Followed**

✅ Reuse of notification channels  
✅ Modular and scalable log-based metric and alerting  
✅ Secrets are securely managed  
✅ Alert documentation included  
✅ Multiple conditions grouped in a single alert policy for simplicity  
✅ All alerts include an auto\_close timeout to reduce stale incidents

**Overview**

This Terraform code sets up a **Google Cloud Monitoring and Alerting infrastructure** for the project **hsbc-12609073-peakplat-dev**. It includes:

* **Log-based metrics** for Kubernetes, Flink, and XDS logs
* **Resource utilization alerts** for VMs, Kubernetes nodes, and Cloud SQL
* **Notification channels** via **xMatters Webhook** and **Email**
* **Alert documentation** using templatefile() for each alert
* **Secrets management** via Google Secret Manager

**☁️ 1. Provider Configuration**

hcl

CopyEdit

provider "google" {

project = var.project\_id

region = var.region

}

* Connects Terraform to GCP using project ID and region variables.

**🔐 2. Secret Manager Integration**

hcl

CopyEdit

data "google\_secret\_manager\_secret\_version" ...

* Securely retrieves **xMatters credentials** (username/password) from **Secret Manager** for use in webhook authentication.

**📢 3. Notification Channels**

**a. xMatters Webhook**

hcl

CopyEdit

resource "google\_monitoring\_notification\_channel" "xmatters\_webhook" ...

* Sends alerts to xMatters using **Basic Auth** and a URL (var.xmatters\_webhook\_url).

**b. Email Notification**

hcl

CopyEdit

resource "google\_monitoring\_notification\_channel" "email" ...

* Sends alert emails to the address specified in var.notification\_email.

**📊 4. Log-Based Metric: Kube Error Logs**

hcl

CopyEdit

resource "google\_logging\_metric" "log\_based\_metrics\_dev" ...

* Creates a log-based metric tracking ERROR logs in the kube-system namespace.

**🚨 5. Alert Policy: Kube Error Logs**

hcl

CopyEdit

resource "google\_monitoring\_alert\_policy" "Log\_Based\_Metrics\_DEV" ...

* Triggers an alert when error logs are detected based on the above log metric.
* Uses:
  + **Condition** on logging.googleapis.com/user/Kube\_Error\_Logs
  + **Threshold**: log count > 0 over 15 minutes
  + **Documentation**: from kube\_error\_logs\_doc.tpl

**🖥️ 6. Alert Policy: VM & K8s Resource Utilization**

hcl

CopyEdit

resource "google\_monitoring\_alert\_policy" "vm\_resource\_utilization\_alert" ...

Includes 3 conditions:

**a. VM Memory > 80%**

* Uses agent.googleapis.com/memory/percent\_used
* 15 min window, threshold: 80%

**b. K8s Node Memory > 50%**

* Uses kubernetes.io/node/memory/allocatable\_utilization
* Threshold: 50%

**c. K8s Node CPU > 50%**

* Uses kubernetes.io/node/cpu/allocatable\_utilization
* Threshold: 50%

**🧠 7. Alert Policy: VM High CPU**

hcl

CopyEdit

resource "google\_monitoring\_alert\_policy" "vm\_high\_cpu\_utilization" ...

* Triggers when agent.googleapis.com/cpu/utilization > 80%
* Uses a **15-minute** window and **mean** alignment

**🧮 8. Alert Policy: Cloud SQL Memory Utilization**

hcl

CopyEdit

resource "google\_monitoring\_alert\_policy" "cloud\_sql\_utilization" ...

* Alerts when Cloud SQL memory usage > 70%
* Uses metric: cloudsql.googleapis.com/database/memory/utilization
* Region filter: europe-west2

**🧮 9. Alert Policy: Cloud SQL CPU Utilization**

hcl

CopyEdit

resource "google\_monitoring\_alert\_policy" "cloud\_sql\_cpu\_utilization" ...

* Triggers when Cloud SQL CPU > 70%
* Uses metric: cloudsql.googleapis.com/database/cpu/utilization

**🔧 10. Log-Based Metric: Flink Logs**

hcl

CopyEdit

resource "google\_logging\_metric" "flink\_log\_alert\_metric" ...

* Captures ERROR logs from Flink pods in Kubernetes.

**🚨 11. Alert Policy: Flink Error Logs**

hcl

CopyEdit

resource "google\_monitoring\_alert\_policy" "flink\_log\_alert\_policy" ...

* Uses the Flink log metric
* Trigger if any error appears within the 10-minute alignment window

**📡 12. Log-Based Metric: XDS Logs**

hcl

CopyEdit

resource "google\_logging\_metric" "xds\_error\_logs\_alerts" ...

* Captures specific log payloads related to XDS issues like RD10014, RD10035, etc.

**🚨 13. Alert Policy: XDS Error Logs**

hcl

CopyEdit

resource "google\_monitoring\_alert\_policy" "xds\_error\_logs\_alerts" ...

* Fires if any of the predefined XDS error messages occur
* Uses a **60-second** alignment period for rapid detection

**📄 14. Documentation with templatefile()**

Each alert uses:

hcl

CopyEdit

documentation {

content = templatefile("${path.module}/alert\_docs/<template>.tpl", {

project\_id = var.project\_id

})

mime\_type = "text/markdown"

}

* Dynamically loads alert documentation content from .tpl templates.
* You can include project-specific variables like ${project\_id} for customization.

**📬 15. Notification Flow**

Each alert sends messages to:

* **xMatters Webhook**: for incident response escalation
* **Email**: for team visibility or logging

**🧼 16. Best Practices Followed**

✅ Reuse of notification channels  
✅ Modular and scalable log-based metric and alerting  
✅ Secrets are securely managed  
✅ Alert documentation included  
✅ Multiple conditions grouped in a single alert policy for simplicity  
✅ All alerts include an auto\_close timeout to reduce stale incidents

**📌 Recommendations for Client**

* These alerts are designed for **early detection** and **incident response**.
* Logs and resource metrics are used in combination to improve observability.
* Easily extendable with more alerts or different services (e.g., BigQuery, Cloud Functions).
* Alerts can be fine-tuned (thresholds/durations) after monitoring real behavior.

Would you like a 1-slide summary deck or talking points for the demo as well?

**You said:**

why we use alert policy

**ChatGPT said:**

We use **alert policies** in GCP Monitoring (and generally in observability) to:

**🚨 Purpose of Alert Policies**

**1. Detect Issues Proactively**

Alert policies notify you **as soon as a problem is detected**, instead of waiting for users or systems to fail noticeably.

Examples:

* High CPU usage on VMs
* Application errors in logs
* Database nearing memory limits

**2. Minimize Downtime**

By responding quickly to alerts, teams can resolve issues **before they affect end users** — improving uptime and reliability.

**3. Automate Incident Response**

Alert policies can trigger:

* Emails
* xMatters incidents (automated on-call routing)
* Slack messages
* PagerDuty notifications  
  ...so teams can **automatically coordinate and escalate** issues.

**4. Support SLO/SLA Compliance**

Alerts help teams meet **Service Level Objectives (SLOs)** or **Service Level Agreements (SLAs)** by ensuring problems are caught within agreed response times.

**5. Enable Data-Driven Operations**

With well-configured alerting, you get:

* Metrics on how often issues happen
* Which systems are most unstable
* How fast your team responds

This allows **continuous improvement** in reliability and operations.

**📈 How It Works (In Your Terraform Code)**

Each alert policy:

* Monitors a **specific condition** (like CPU > 80% for 15 mins)
* Is based on a **metric or log-based metric**
* Defines **notification channels** (xMatters, email)
* Includes **documentation** to guide responders
* Optionally auto-closes after X minutes if the condition resolves

**🔧 Without Alert Policies...**

* You’d have to constantly check dashboards manually.
* You’d miss real-time failures.
* MTTR (mean time to resolution) would increase.
* SLA violations would go undetected.

**🧠 In Summary:**

"**Alert policies are your system’s immune response — they detect problems, raise flags, and help fix issues quickly, before they spread.**"

Would you like a short version for client slides or speech notes?