{ DevFest }

Lucknow

Harnessing Google Cloud for Real-Time Problem Solving through Observability

Google Developer Groups

Experts

Hello!

I am Saurabh Mishra

DevOps Lead working with TSYS (Global Payments)
Got Bachelors, degrees in Information Technology
GDE- GCP and Champion Innovator
DevOps Institute Ambassador and Organizer
AWS CB||Calico Big Cats ||CDF AMB|| open-appsec AMB

Feel free to follow me at:

LinkedIn (www.linkedin.com/connectsaurabhmishra)
Medium (www.medium.com/@connectsaurabhmishra)

Experts

Observability | Rules

Questions are welcome at any time

Please silence your phone

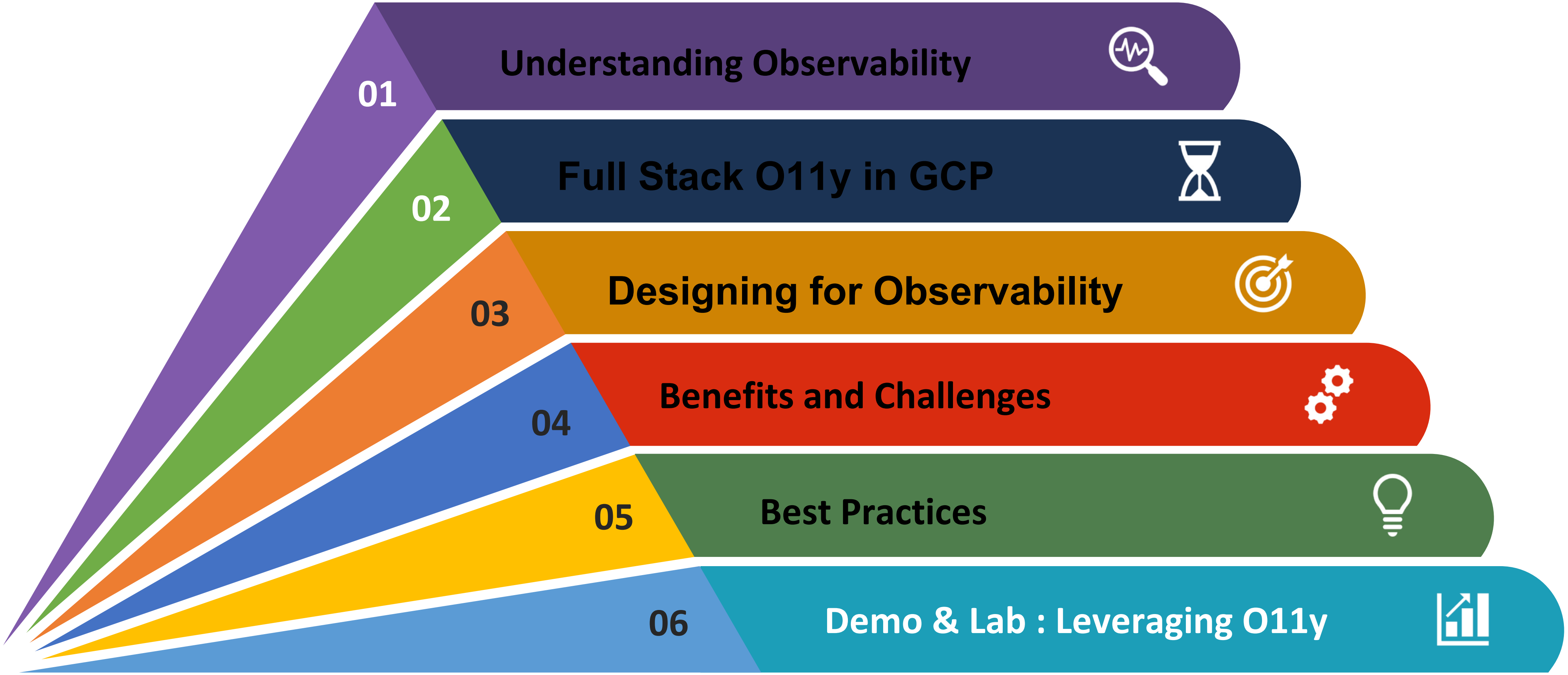
There are NO STUPID Questions

Most “stupid” ones are the most welcome

Keep the session INTERACTIVE

Share experiences, stories and build sustainable working practices

Observability | Agenda



01 Understanding Observability

02 Full Stack O11y in GCP

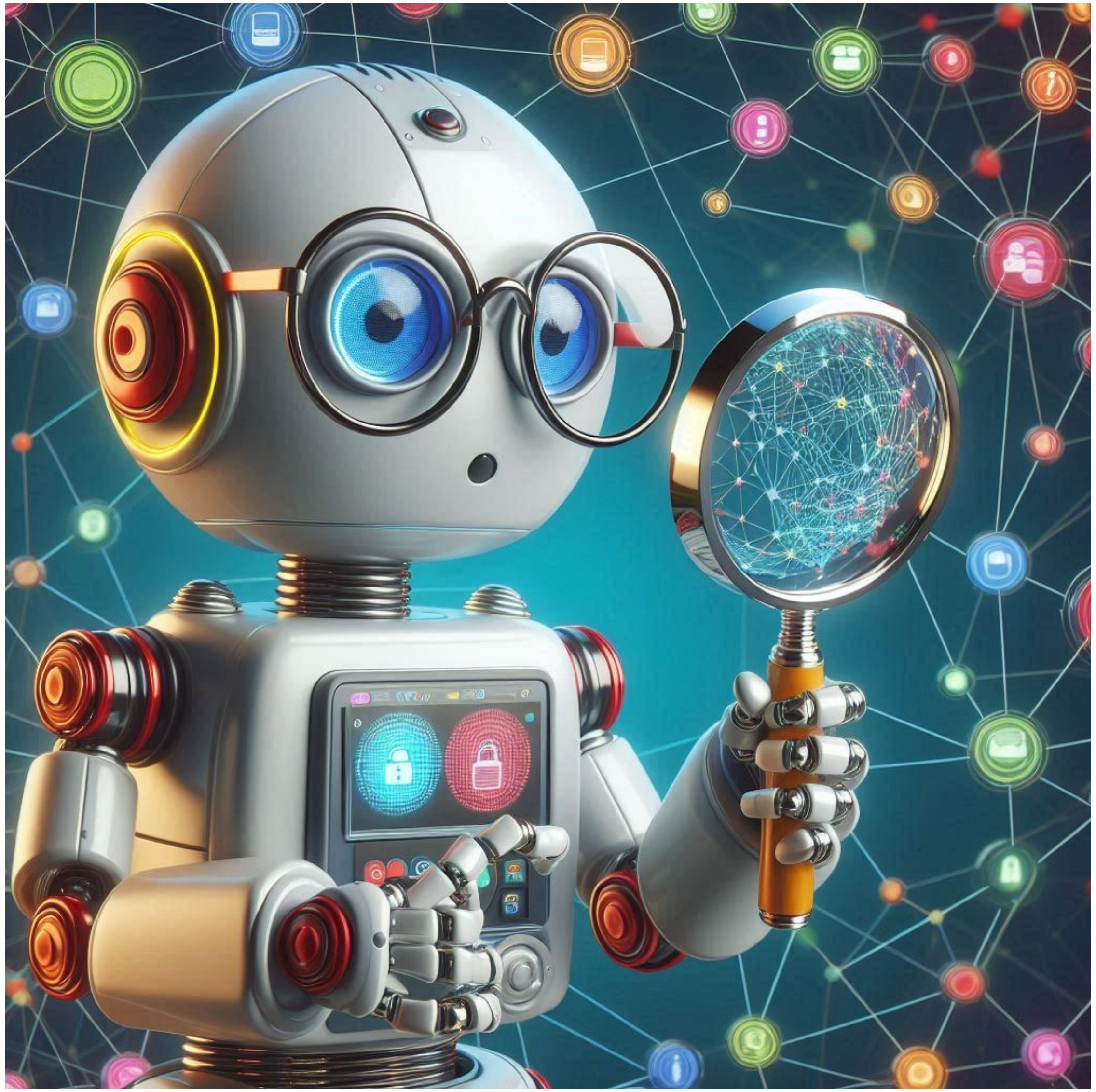
03 Designing for Observability

04 Benefits and Challenges

05 Best Practices

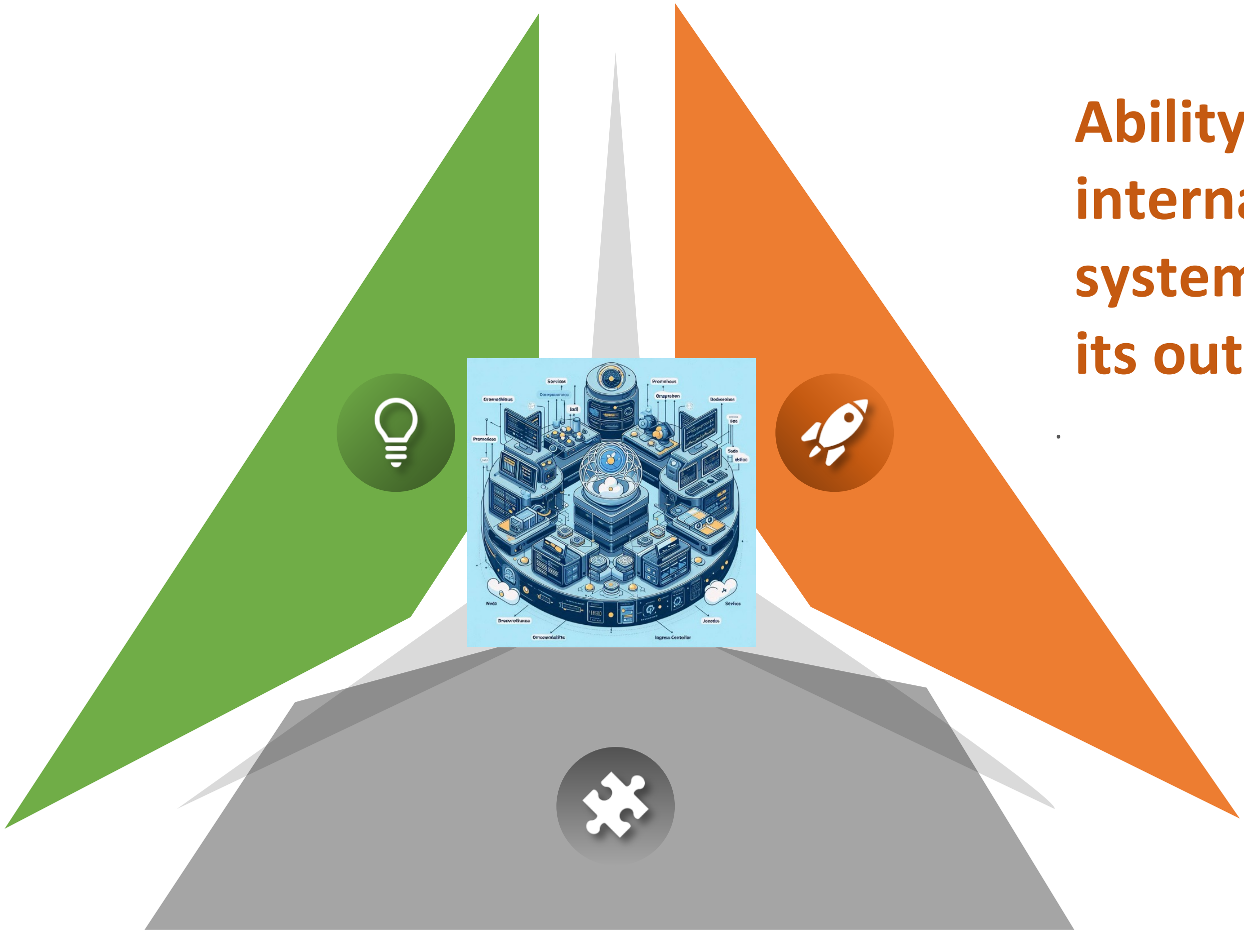
06 Demo & Lab : Leveraging O11y

Observability | Definition



Experts

Observability | Definition

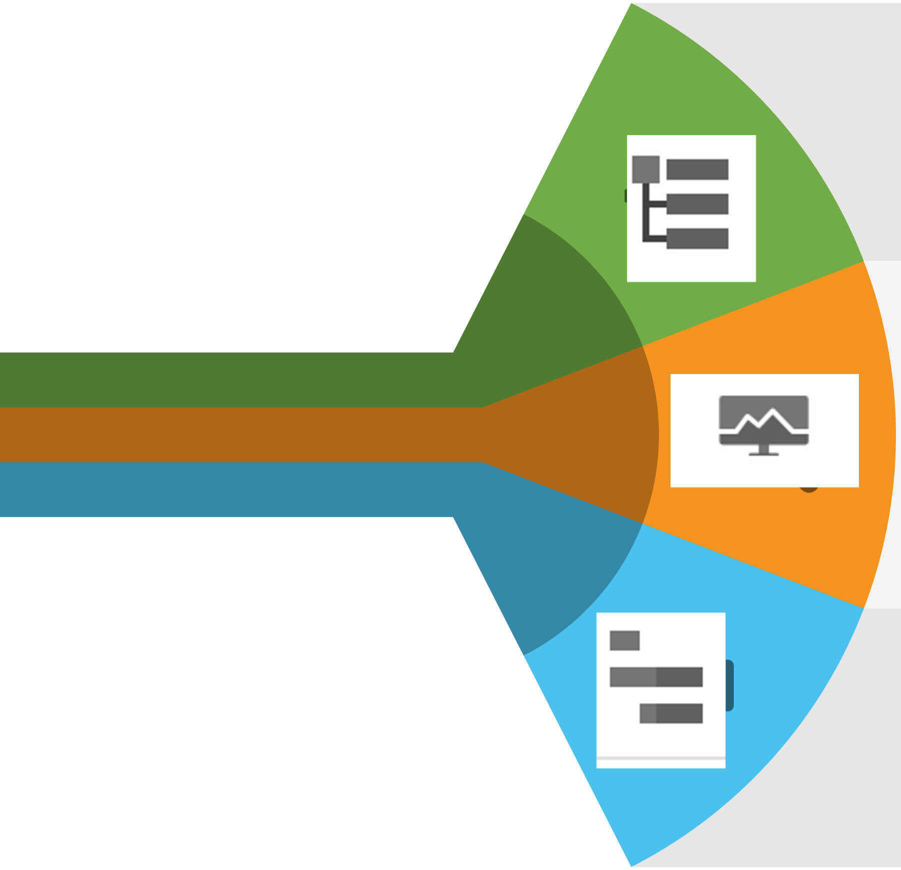


Ensures visibility into complex, distributed environments for better system management and optimization

Ability to measure the internal states of a system by examining its outputs

Gaining insight into the behavior and performance of running applications

Observability | Pillars



Logs

Explain Why it is happening

Metrics

Tell us what is happening

Traces

Shows how it is happening

Observability | Chaos Engineering

Concept of "Monkey" refers to a set of tools or services that simulate various failures and disruptions in a system to test its resilience and stability

Observability | Chaos Engineering

Deliberately introducing controlled failures into the system

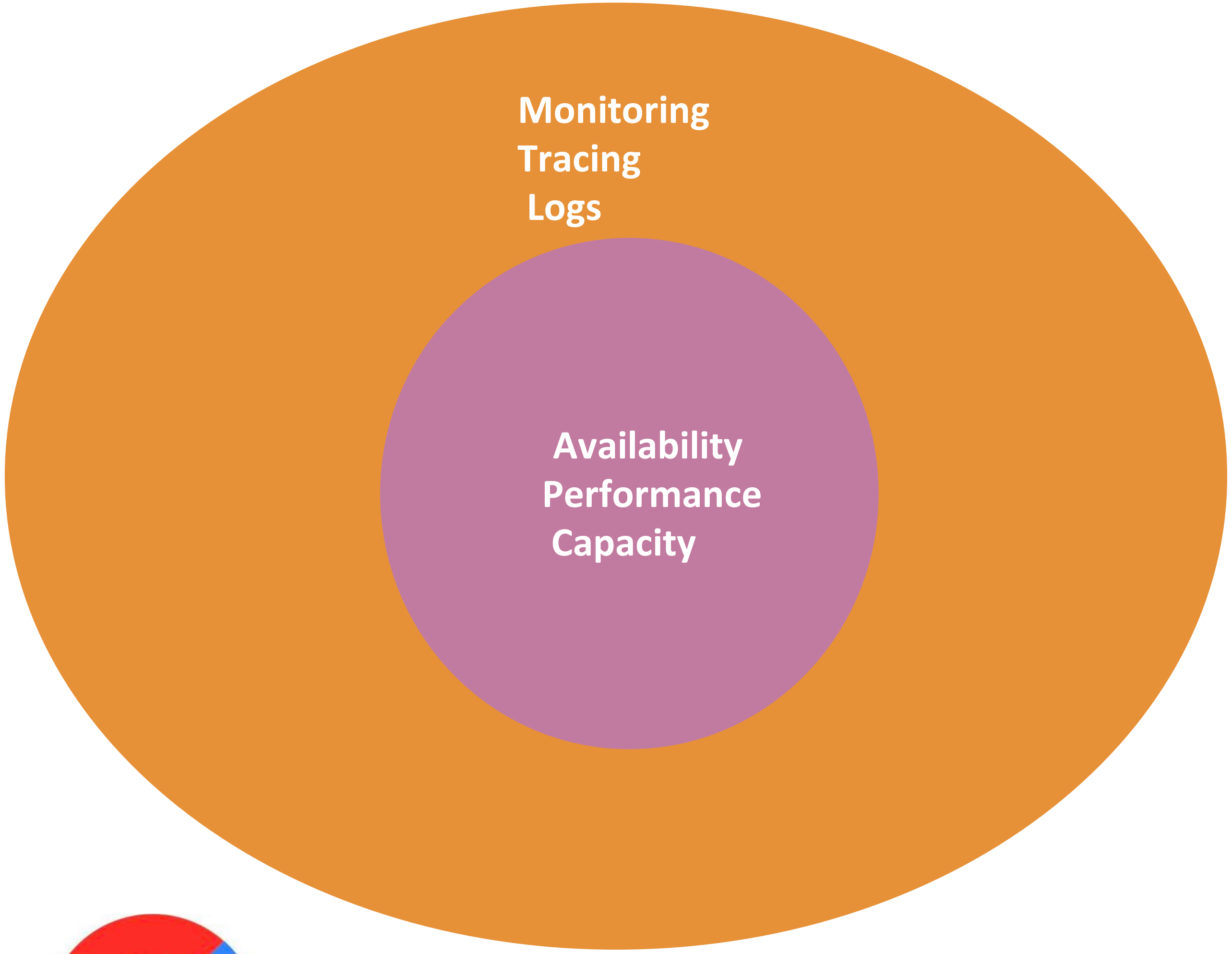
Intentional Failure Injection :- Introducing disruptions such as pod failures, node outages, network latency, or CPU spikes.

Monitoring System Behavior: Observing how applications respond to failures and ensuring they can recover gracefully.

Learning from Failures: Analyzing the results of chaos experiments to identify potential areas of improvement.

Observability | Monitoring v Observability

Observability |Monitoring vs Observability



Monitoring
Tracing
Logs

Availability
Performance
Capacity

Aspect	Monitoring	Observability
Primary Goals	Identify and alert on known issues and metrics (When & What)	Understand and explore unknown and emergent behaviors (Why & How)
Scope	Primarily focused on uptime and system metrics	Encompasses logs, metrics, traces, and more to provide a holistic view
Data Sources	Metrics (e.g., CPU usage, memory, response time)	Metrics, logs, traces, events, and all output data sources
Approach	Pre-configured dashboards and alerting rules	Supports ad hoc querying and real-time analysis
Nature	Reactive	Proactive
Complexity	Easier to implement with standard metrics and thresholds	More complex, requires a deep integration for comprehensive data capture
Example Tools	Nagios, Zabbix, Prometheus (metrics-focused)	Grafana, OpenTelemetry, Jaeger
Use Case	Operators and IT teams monitoring uptime and performance	Investigate why response times have increased without a predefined alert

Experts

Observability | Monitoring v Observability

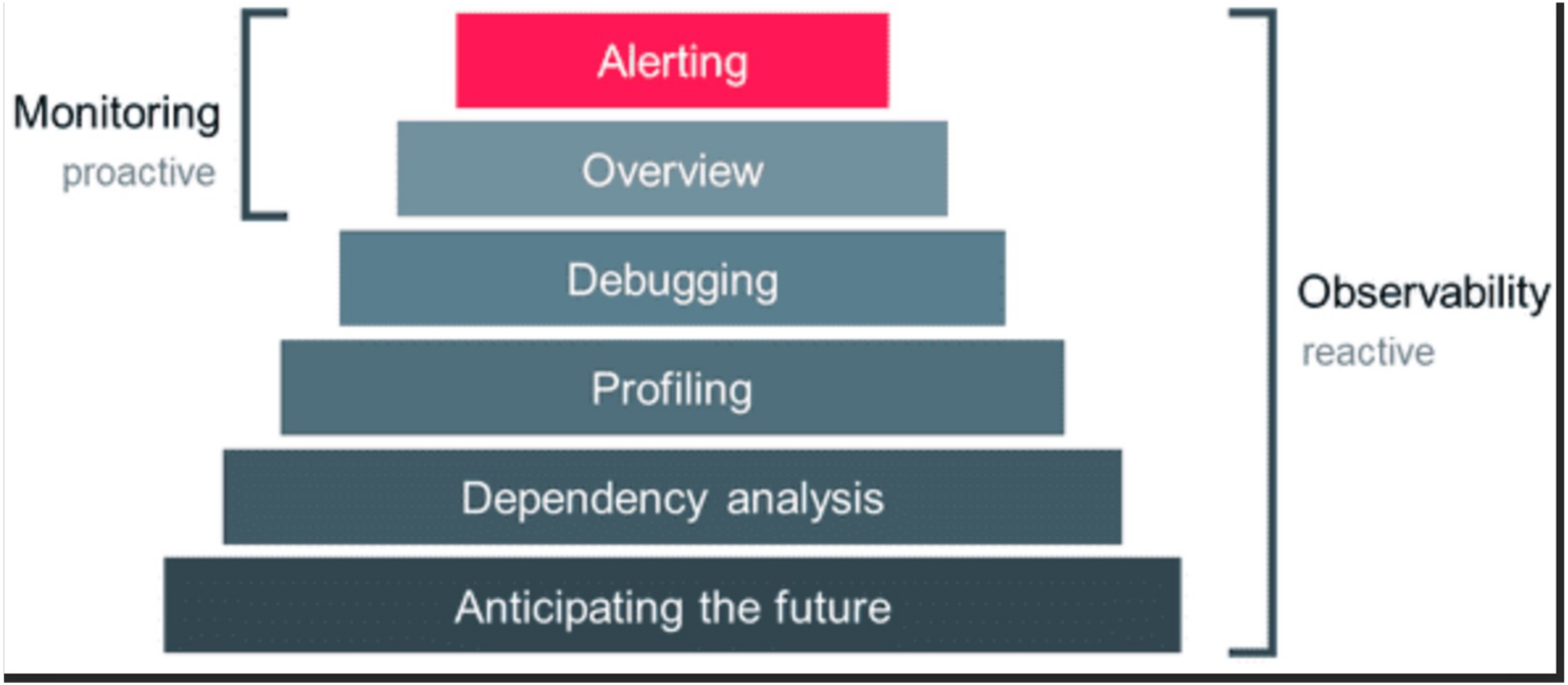
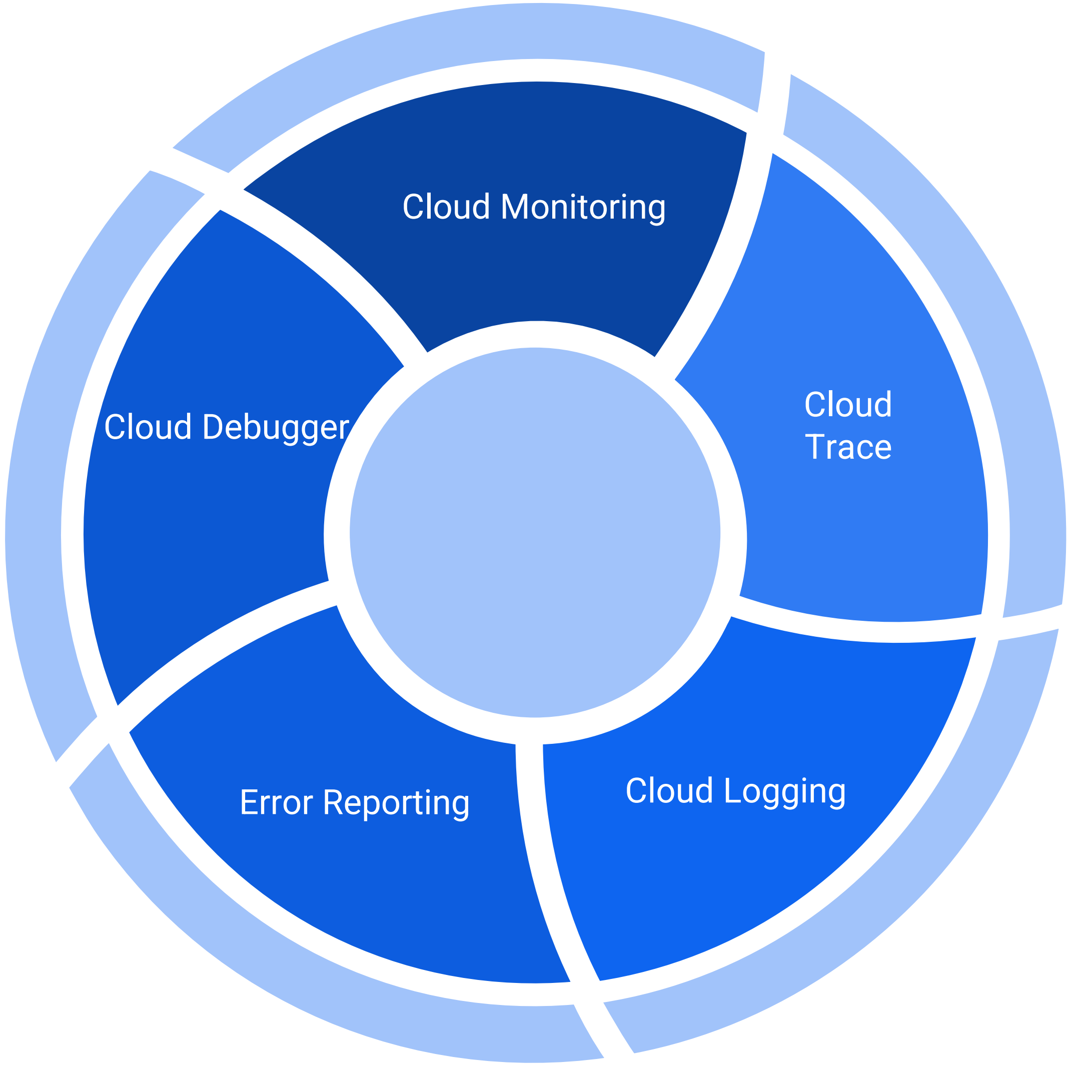
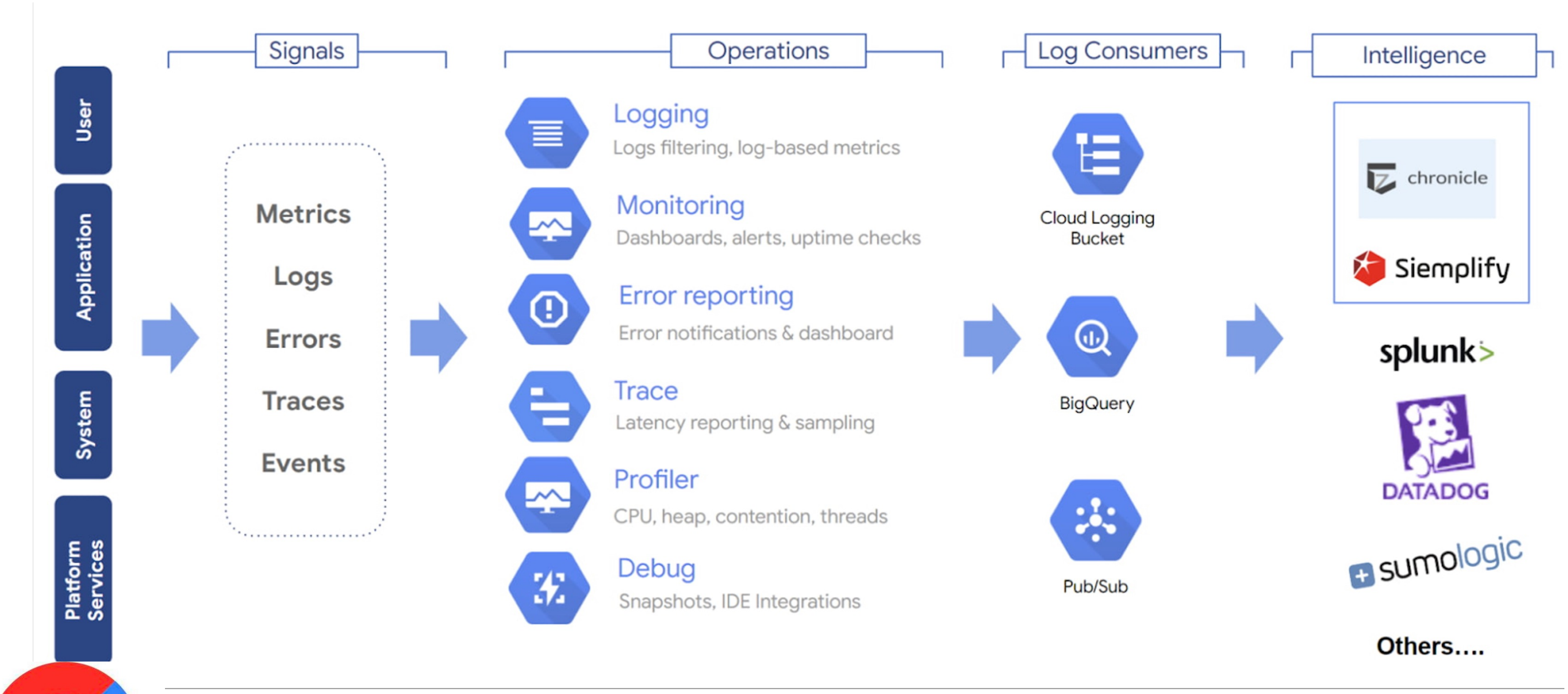


Image sourced from Google Search

Observability | Google Cloud's Operations Suite in GCP



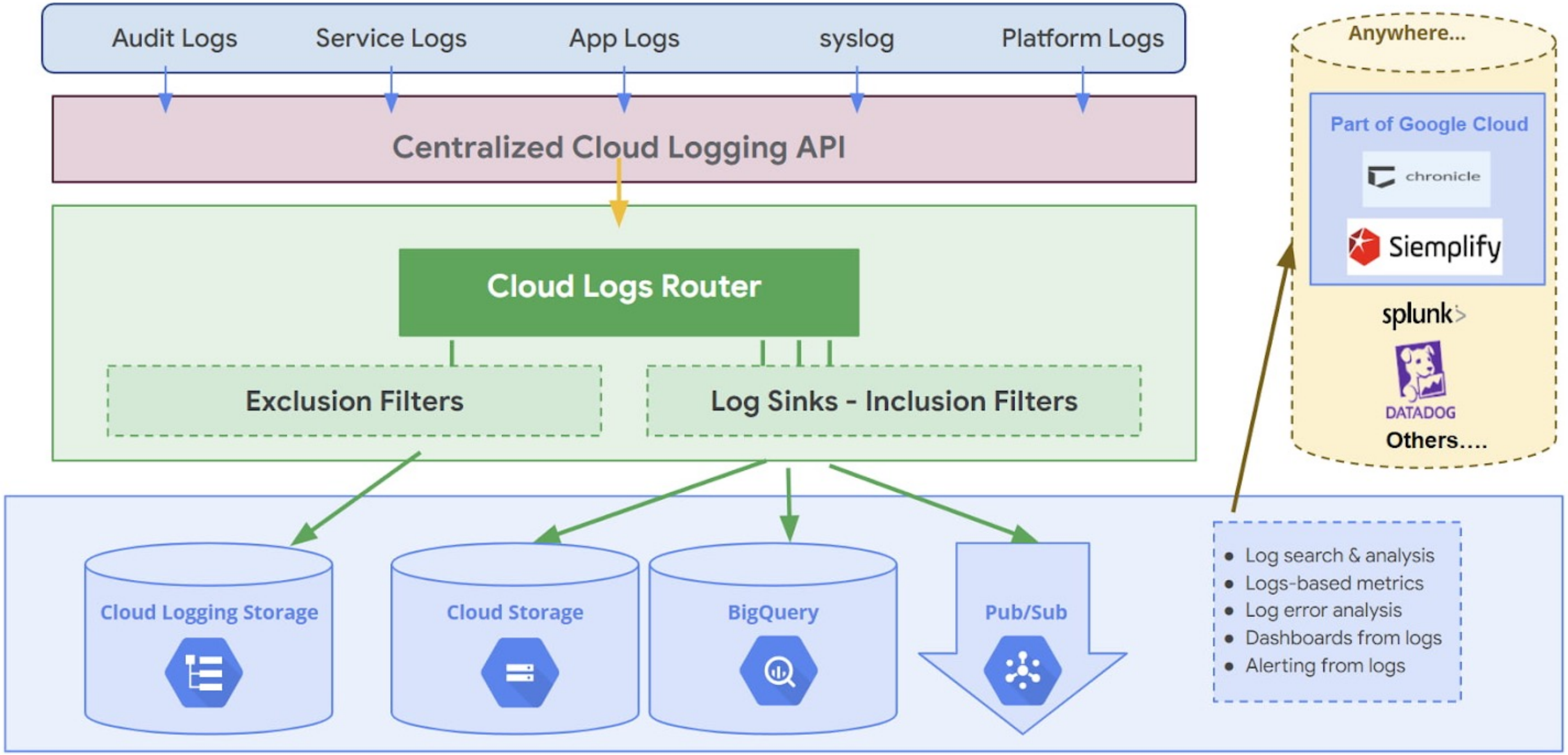
Observability | Google Cloud's operations suite in GCP



Experts

https://cloud.google.com/blog/products/management-tools/observability-on-google-cloud

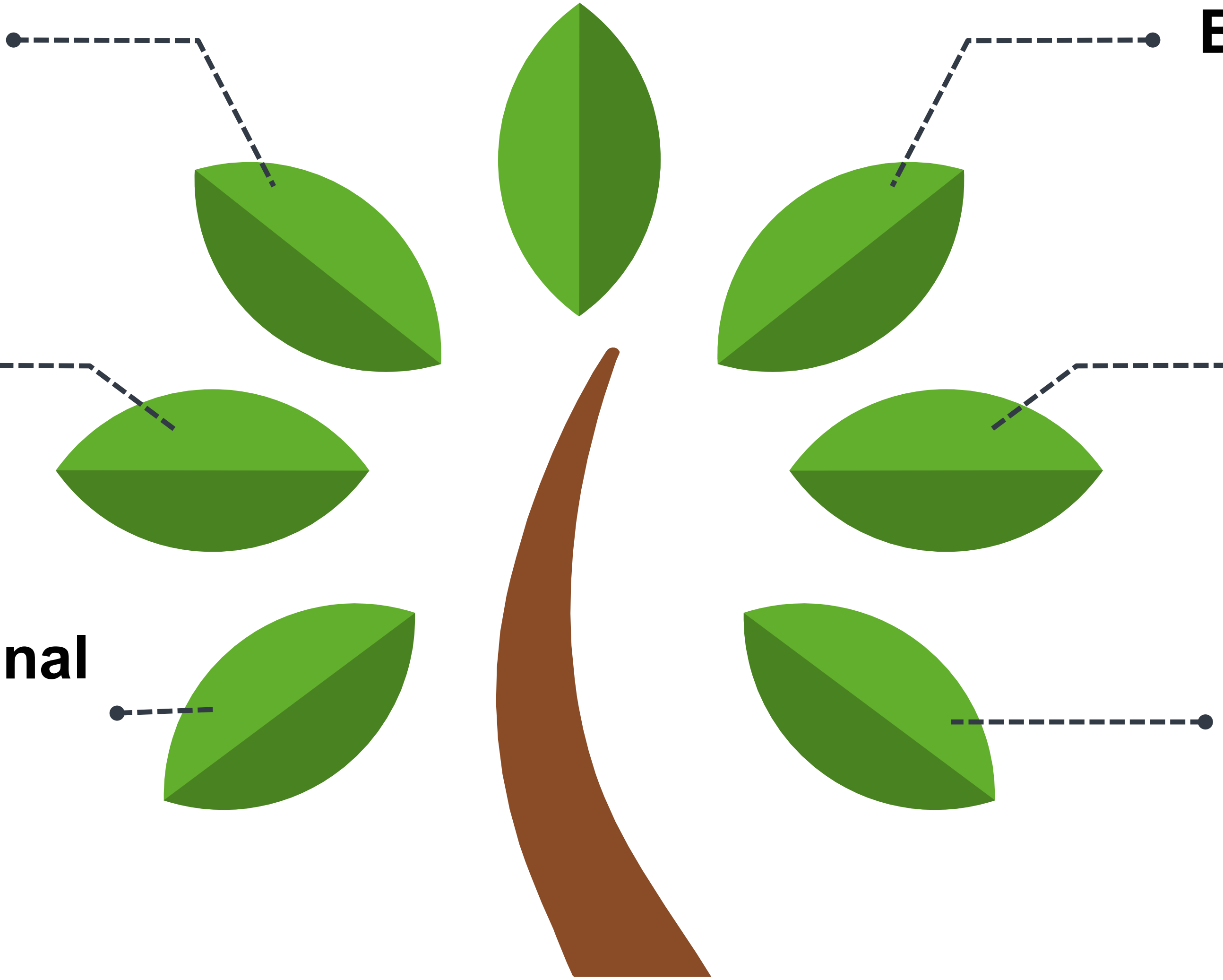
Observability | Google Cloud's operations suite in GCP



- Log search & analysis
- Logs-based metrics
- Log error analysis
- Dashboards from logs
- Alerting from logs

https://cloud.google.com/blog/products/management-tools/observability-on-google-cloud

Observability | Benefits



Improve System reliability

Better User Experience

Optimize Operational Cost

Better Visibility

Better Workflow

Faster Alerting & troubleshooting

Observability | Challenges

Data Silos

Data Overload and Volume

Lack of pre-production

Tool Fragmentation & Integration

Alert Fatigue & Wasting time troubleshooting

Observability | Best Practices

Best Practices

Leverage Logging Best Practices

Understand your environment

Automate tasks through various tools

Multi-Layered Monitoring

Don’t prefer to monitor everything's

Observability | Lab & Demo

How to collect and view latency data from applications

Create a Google Kubernetes Engine (GKE) cluster by using the Google Cloud CLI.

```
gcloud container clusters create cloud-trace-demo --zone us-central1-c

gcloud container clusters get-credentials cloud-trace-demo --zone us-central1-c
```

Download and deploy a sample application to cluster.

```
git clone https://github.com/GoogleCloudPlatform/python-docs-samples.git

cd python-docs-samples/trace/cloud-trace-demo-app-opentelemetry && ./setup.sh
```

Create a trace by sending an HTTP request to the sample application.

```
curl $(kubectl get svc -o=jsonpath='{.items[?(@.metadata.name=="cloud-trace-demo-a")].status.loadBalancer.ingress[0].ip}’)
```

View the latency information of the trace you created.

Experts

Observability | Lab & Demo

Experts

How to collect and view latency data from applications

Create a Google Kubernetes Engine (GKE) cluster by using the Google Cloud CLI.

```
gcloud container clusters create cloud-trace-demo --zone us-central1-c

gcloud container clusters get-credentials cloud-trace-demo --zone us-central1-c
```

Download and deploy a sample application to cluster.

```
git clone https://github.com/GoogleCloudPlatform/python-docs-samples.git

cd python-docs-samples/trace/cloud-trace-demo-app-opentelemetry && ./setup.sh
```

Create a trace by sending an HTTP request to the sample application.

```
curl $(kubectl get svc -o=jsonpath='{.items[?(@.metadata.name=="cloud-trace-demo-a")].status.loadBalancer.ingress[0].ip}’)
```

View the latency information of the trace you created.

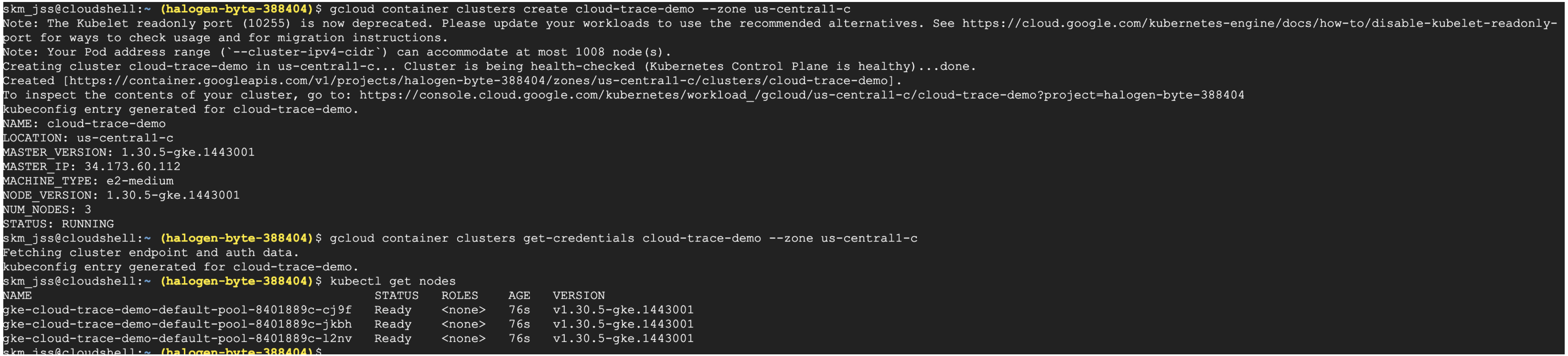
Observability | Lab & Demo (Contd.)

How to collect and view latency data from applications

Create a Google Kubernetes Engine (GKE) cluster by using the Google Cloud CLI.

gcloud container clusters create cloud-trace-demo --zone us-central1-c

gcloud container clusters get-credentials cloud-trace-demo --zone us-central1-c



```
skm_jss@cloudshell:~ (halogen-byte-388404)$ gcloud container clusters create cloud-trace-demo --zone us-central1-c
Note: The Kubelet readonly port (10255) is now deprecated. Please update your workloads to use the recommended alternatives. See https://cloud.google.com/kubernetes-engine/docs/how-to/disable-kubelet-readonly-
port for ways to check usage and for migration instructions.
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster cloud-trace-demo in us-central1-c... Cluster is being health-checked (Kubernetes Control Plane is healthy)...done.
Created [https://container.googleapis.com/v1/projects/halogen-byte-388404/zones/us-central1-c/clusters/cloud-trace-demo].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/us-central1-c/cloud-trace-demo?project=halogen-byte-388404
kubeconfig entry generated for cloud-trace-demo.
NAME: cloud-trace-demo
LOCATION: us-central1-c
MASTER_VERSION: 1.30.5-gke.1443001
MASTER_IP: 34.173.60.112
MACHINE_TYPE: e2-medium
NODE_VERSION: 1.30.5-gke.1443001
NUM_NODES: 3
STATUS: RUNNING
skm_jss@cloudshell:~ (halogen-byte-388404)$ gcloud container clusters get-credentials cloud-trace-demo --zone us-central1-c
Fetching cluster endpoint and auth data.
kubeconfig entry generated for cloud-trace-demo.
skm_jss@cloudshell:~ (halogen-byte-388404)$ kubectl get nodes
NAME                                              STATUS   ROLES    AGE   VERSION
gke-cloud-trace-demo-default-pool-8401889c-cj9f   Ready    <none>   76s   v1.30.5-gke.1443001
gke-cloud-trace-demo-default-pool-8401889c-jkbh   Ready    <none>   76s   v1.30.5-gke.1443001
gke-cloud-trace-demo-default-pool-8401889c-l2nv   Ready    <none>   76s   v1.30.5-gke.1443001
skm_jss@cloudshell:~ (halogen-byte-388404)$
```

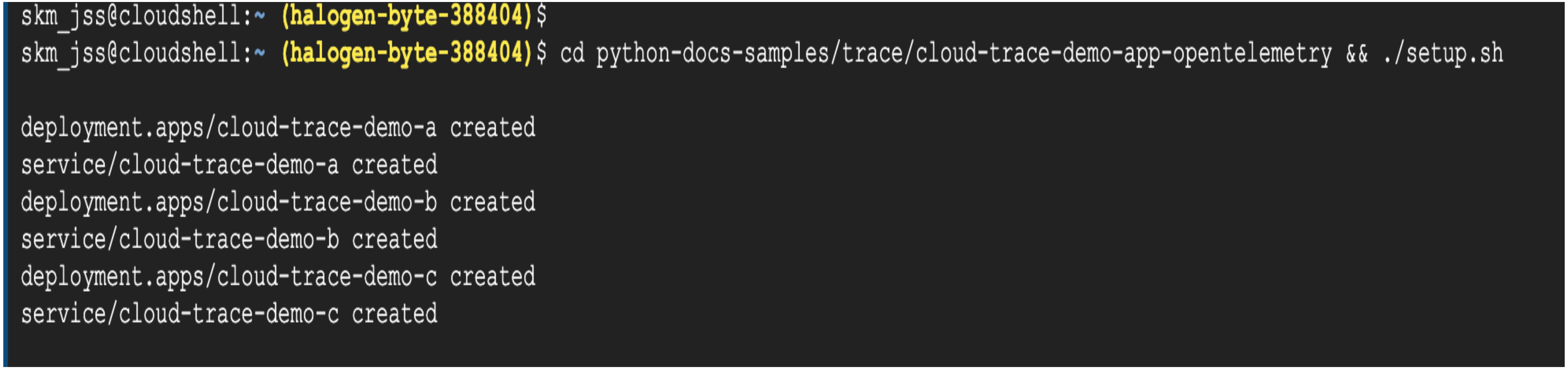
Observability | Lab & Demo (Contd.)

How to collect and view latency data from applications

Download and deploy a sample application to cluster.

git clone https://github.com/GoogleCloudPlatform/python-docs-samples.git

cd python-docs-samples/trace/cloud-trace-demo-app-opentelemetry && ./setup.sh



```
skm_jss@cloudshell:~ (halogen-byte-388404)$
skm_jss@cloudshell:~ (halogen-byte-388404)$ cd python-docs-samples/trace/cloud-trace-demo-app-opentelemetry && ./setup.sh

deployment.apps/cloud-trace-demo-a created
service/cloud-trace-demo-a created
deployment.apps/cloud-trace-demo-b created
service/cloud-trace-demo-b created
deployment.apps/cloud-trace-demo-c created
service/cloud-trace-demo-c created
```

Observability | Lab & Demo (Contd.)

How to collect and view latency data from applications

Create a trace by sending an HTTP request to the sample application.

curl $(kubectl get svc -o=jsonpath='{.items[?(@.metadata.name=="cloud-trace-demo-a")].status.loadBalancer.ingress[0].ip}')

kubectl fetches the IP address of the service named cloud-trace-demo-a.

The curl command then sends the HTTP request to service a.

Service a receives the HTTP request and sends a request to service b

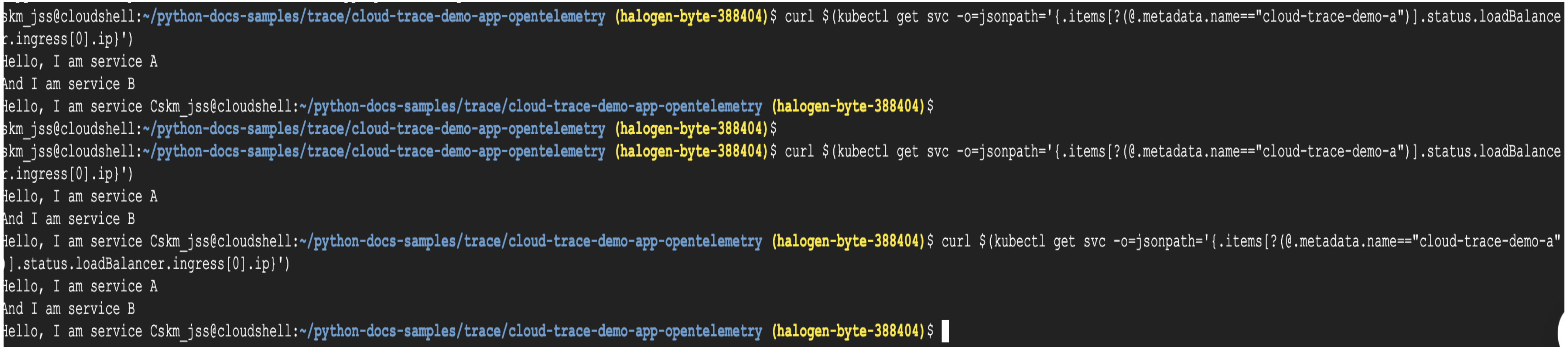
Service b receives the HTTP request and sends a request to service c.

Service c receives the HTTP request from service b and returns the string Hello, I am service C to service b.

Service b receives the response from service c, appends it to the string And I am service B, and returns the result to service a.

Service a receives the response from service b and appends it to the string Hello, I am service A.

The response from service a is printed in the Cloud Shell.

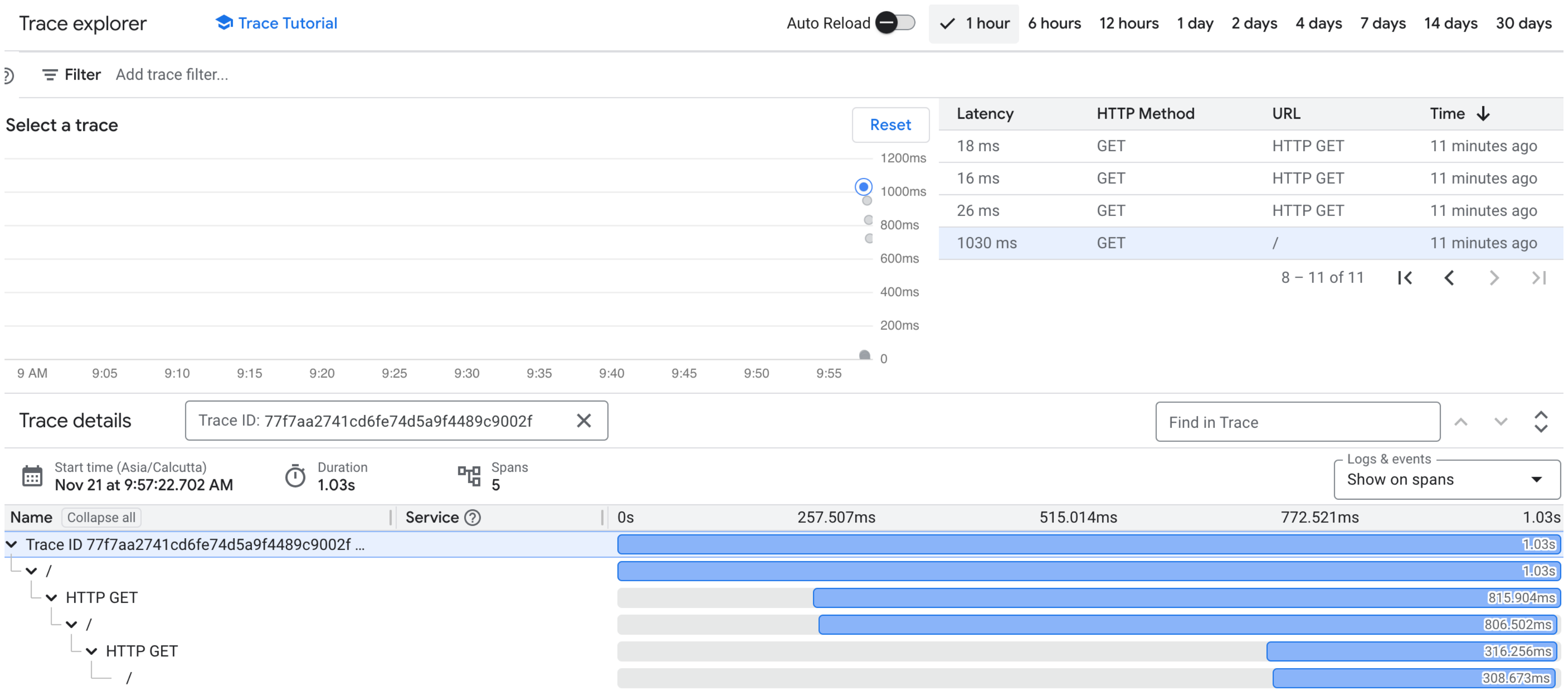


```
skm_jss@cloudshell:~/python-docs-samples/trace/cloud-trace-demo-app-opentelemetry (halogen-byte-388404)$ curl $(kubectl get svc -o=jsonpath='{.items[?(@.metadata.name=="cloud-trace-demo-a")].status.loadBalance
r.ingress[0].ip}')
Hello, I am service A
And I am service B
Hello, I am service Cskm_jss@cloudshell:~/python-docs-samples/trace/cloud-trace-demo-app-opentelemetry (halogen-byte-388404)$
skm_jss@cloudshell:~/python-docs-samples/trace/cloud-trace-demo-app-opentelemetry (halogen-byte-388404)$
skm_jss@cloudshell:~/python-docs-samples/trace/cloud-trace-demo-app-opentelemetry (halogen-byte-388404)$ curl $(kubectl get svc -o=jsonpath='{.items[?(@.metadata.name=="cloud-trace-demo-a")].status.loadBalance
r.ingress[0].ip}')
Hello, I am service A
And I am service B
Hello, I am service Cskm_jss@cloudshell:~/python-docs-samples/trace/cloud-trace-demo-app-opentelemetry (halogen-byte-388404)$ curl $(kubectl get svc -o=jsonpath='{.items[?(@.metadata.name=="cloud-trace-demo-a"
)].status.loadBalancer.ingress[0].ip}')
Hello, I am service A
And I am service B
Hello, I am service Cskm_jss@cloudshell:~/python-docs-samples/trace/cloud-trace-demo-app-opentelemetry (halogen-byte-388404)$
```

Observability | Lab & Demo (Contd.)

How to collect and view latency data from applications

View the latency information of the trace you created.



Latency	HTTP Method	URL	Time ↓
18 ms	GET	HTTP GET	11 minutes ago
16 ms	GET	HTTP GET	11 minutes ago
26 ms	GET	HTTP GET	11 minutes ago
1030 ms	GET	/	11 minutes ago

8 – 11 of 11

Trace details

Trace ID: 77f7aa2741cd6fe74d5a9f4489c9002f

Start time (Asia/Calcutta) Nov 21 at 9:57:22.702 AM

Duration 1.03s

Spans 5

Logs & events: Show on spans

Name (Collapse all)	Service	0s · 257.507ms · 515.014ms · 772.521ms · 1.03s
Trace ID 77f7aa2741cd6fe74d5a9f4489c9002f ...		1.03s
/		1.03s
HTTP GET		815.904ms
/		806.502ms
HTTP GET		316.256ms
/		308.673ms

Observability | References

- Observability: - Beyond monitoring & real time problem solving on Google Cloud | by Saurabh Mishra | Google Cloud - Community | Jun, 2023 | Medium
- Operations: Cloud Monitoring & Logging | Google Cloud
- SKILup IT Learning — DevOps Institute
- Introduction to Google Cloud's operations suite | Google Cloud Blog
- Observability: 3 things about it. | Isham Araia's Blog (ish-ar.io)
- Observability vs. monitoring: What’s the difference? (dynatrace.com)
- Why Distributed Tracing is Essential for APM | New Relic
- View a trace | Cloud Trace | Google Cloud
- The 3 pillars of observability: Logs, metrics and traces | TechTarget
- https://landscape.cncf.io/

Experts

Thank You !

Experts