Logging and Alerting

Cloud Security

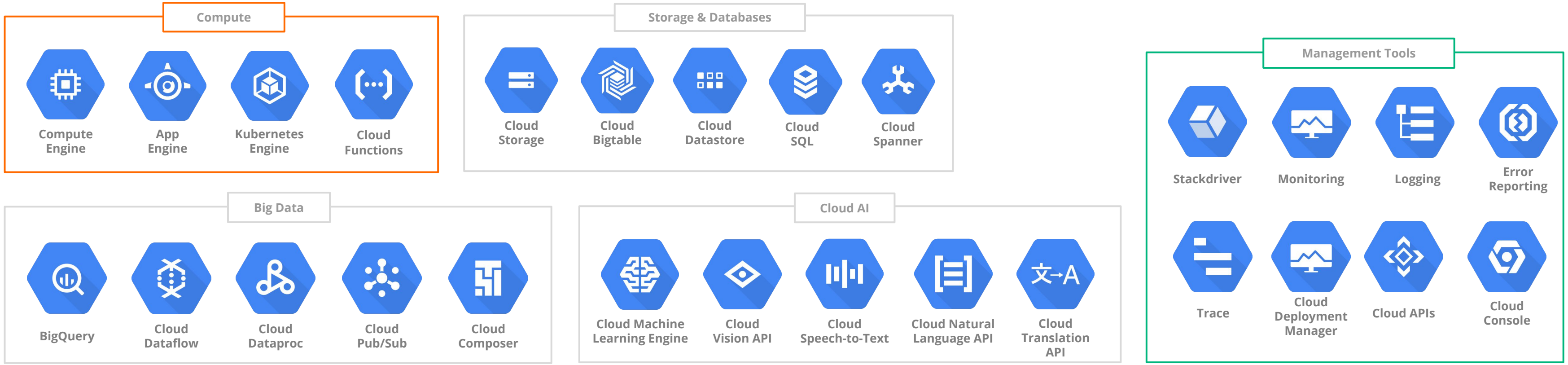
About me

- Senior Security Engineer at King
- Telecommunications Engineer
- Network and Network Security Background
- Native from Romania
- @rorutza

Content

- GCP Overview
- The Problem
- Logging and Telemetry
- Alerting and Use Cases
- Automation
    - Google Cloud Functions
- Conclusions

Google Cloud Platform



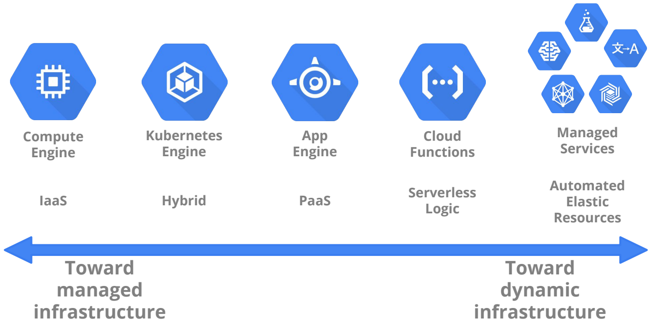
Compute
- Compute Engine
- App Engine
- Kubernetes Engine
- Cloud Functions

Storage & Databases
- Cloud Storage
- Cloud Bigtable
- Cloud Datastore
- Cloud SQL
- Cloud Spanner

Management Tools
- Stackdriver
- Monitoring
- Logging
- Error Reporting
- Trace
- Cloud Deployment Manager
- Cloud APIs
- Cloud Console

Big Data
- BigQuery
- Cloud Dataflow
- Cloud Dataproc
- Cloud Pub/Sub
- Cloud Composer

Cloud AI
- Cloud Machine Learning Engine
- Cloud Vision API
- Cloud Speech-to-Text
- Cloud Natural Language API
- Cloud Translation API



Compute Engine	Kubernetes Engine	App Engine	Cloud Functions	Managed Services
IaaS	Hybrid	PaaS	Serverless Logic	Automated Elastic Resources

Toward managed infrastructure ↔ Toward dynamic infrastructure

King collaborates with Google Cloud for next-generation analytics and machine learning

Google Cloud

King

The Problem

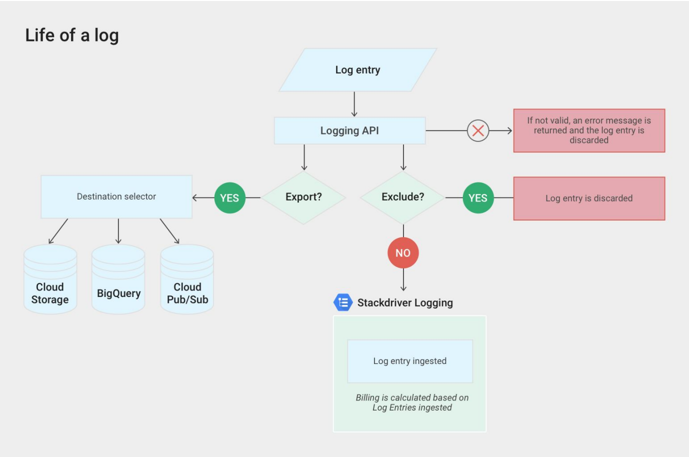
- New uncharted environment
- On-prem mentality
- Lack of visibility
- User independence
- Feeling of losing control
- Data leak news

How do we solve it?

- GCP native tools
- Community
- Practice
- Study

Logging and Telemetry

- Google Stackdriver
- Sinks and Exports
  - Queries



Life of a log

Log entry

Logging API

If not valid, an error message is returned and the log entry is discarded

Export?

YES

Destination selector

Cloud Storage

BigQuery

Cloud Pub/Sub

Exclude?

YES

Log entry is discarded

NO

Stackdriver Logging

Log entry ingested

Billing is calculated based on Log Entries ingested

- Cloud Audit Logging
  - Cloud Audit Logs
  - Data Access Logs
  - System Activity
- Telemetry
  - Load Balancer Telemetry
  - VPC Flows
  - FW Logs
- Application Specific
  - AppEngine Access Logs
  - BigQuery Logs

Admin Activity

- Activity performed by users over resources
  - Resource creation, permissions, etc.
- Information
  - User

    `protoPayload.authenticationInfo.principalEmail`
  - Source IP

    `protoPayload.requestMetadata.callerIp`
  - Method name

    `protoPayload.methodName`
  - Resource name

    `protoPayload.resourceName`
  - Project name

    `resource.labels.project_id`

- IAM change fields

```
protoPayload.serviceData.policyDelta.bindingDeltas.member
protoPayload.serviceData.policyDelta.bindingDeltas.role
```

- VPC Firewall Rules

```
protoPayload.request.alloweds.ports
protoPayload.request.direction
protoPayload.request.sourceRanges
protoPayload.request.targetTags
```

- Compute Engine VM Instance

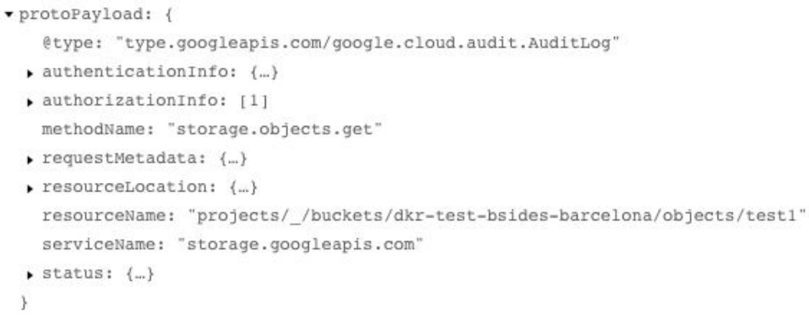
```
resource.type = "gce_instance"
resource.labels.instance_id ="{#instance_id}"
logName  =
"projects/{#project_id}/logs/cloudaudit.googleapis.com
%2Factivity"
```

Data Access

- Data accessed by user
  - Not enabled by default
  - API calls that create, modify or read user-provided data
    - Which users and accounts performed various GCP calls/actions?
    - When/where the calls occurred?
    - Who called/made them?
- Information
  - User
  - methodName
  - resourceName
  - projectName

- Data Access logs for a single GCS Bucket by a user

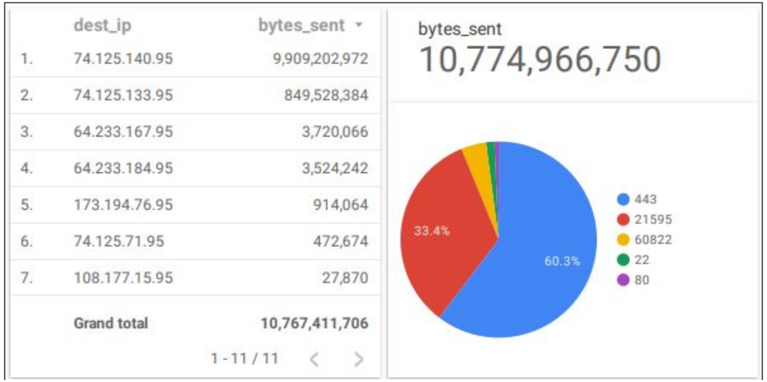
```
resource.type = ”gcs_bucket"
resource.labels.bucket_name ="{#bucket_name}"
authenticationInfo.principalEmail ="{#email}"
logName  =
"projects/{#project_id}/logs/cloudaudit.googleapis.com
%2F data_access"
```



```
▾ protoPayload: {
    @type: "type.googleapis.com/google.cloud.audit.AuditLog"
  ▸ authenticationInfo: {…}
  ▸ authorizationInfo: [1]
    methodName: "storage.objects.get"
  ▸ requestMetadata: {…}
  ▸ resourceLocation: {…}
    resourceName: "projects/_/buckets/dkr-test-bsides-barcelona/objects/test1"
    serviceName: "storage.googleapis.com"
  ▸ status: {…}
  }
```

VM Instance Traffic

- Connection
  - `src_ip, src_port, dest_ip, det_port, protocol`
- Traffic volume
  - `bytes_sent, packets_sent`
- VPC Network Details
  - `project_id, vpc_name, subnetwork_name`



	dest_ip	bytes_sent ▾
1.	74.125.140.95	9,909,202,972
2.	74.125.133.95	849,528,384
3.	64.233.167.95	3,720,066
4.	64.233.184.95	3,524,242
5.	173.194.76.95	914,064
6.	74.125.71.95	472,674
7.	108.177.15.95	27,870
	Grand total	10,767,411,706

1 - 11 / 11

bytes_sent

10,774,966,750

- Traffic for a specific VM

```
resource.type="gce_subnetwork"
logName="projects/{#project_id}/logs/compute.googleapis.com%2Fvpc_flows"
jsonPayload.src_instance.vm_name="{#vm_name}”
```

- Trafic for a specific port and protocol

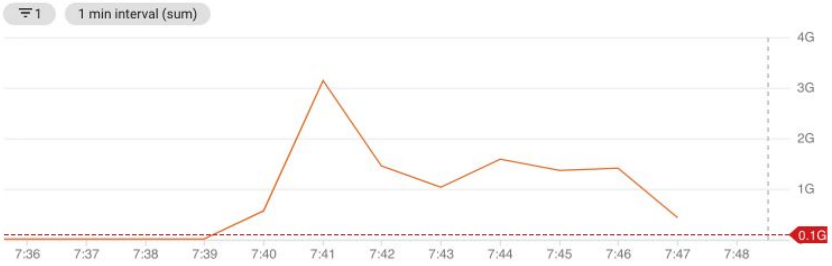
```
resource.type="gce_subnetwork"
logName="projects/{#project_id}/logs/compute.googleapis.com%2Fvpc_flows"
jsonPayload.src_instance.vm_name="{#vm_name}”
```

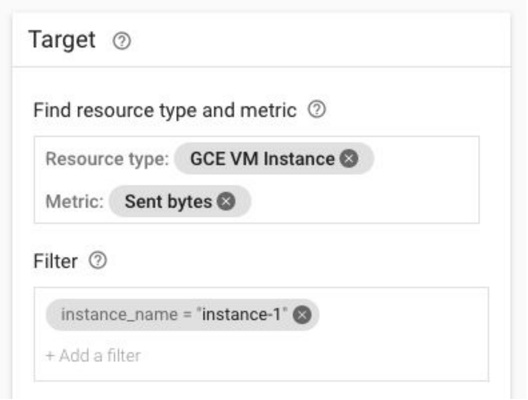
- Traffic for a specific subnet

```
resource.type="gce_subnetwork"
logName="projects/{#project_id}/logs/compute.googleapis.com%2Fvpc_flows"
ip_in_net(jsonPayload.connection.dest_ip, {#subnet})
```

Alerts

- Stackdriver
  - Exfil from VM
  - High CPU load
  - User-created metrics
- Based on logging
  - Firewall and VPC flow changes
  - IAM changes on selected projects
  - Creation of non compliant VM
  - High resource consumption
  - Non-domain account accessing GCP
  - Traffic volume alerts




Target

Find resource type and metric

Resource type: GCE VM Instance

Metric: Sent bytes

Filter

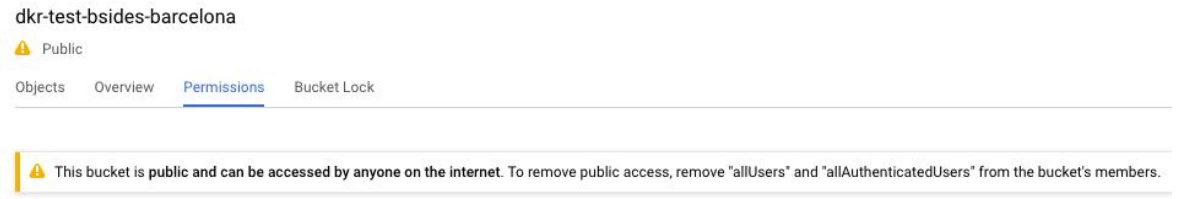
instance_name = "instance-1"

+ Add a filter

Public Bucket

- Special member identifiers
  - allUsers
  - allAuthenticatedUsers

- Alert created in Stackdriver
  - User-Defined Metric
  - Alert Policy based on the metric



dkr-test-bsides-barcelona

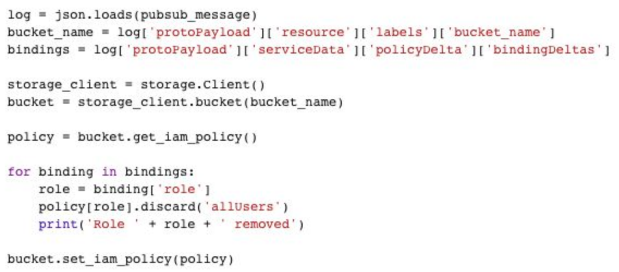
⚠ Public

⚠ This bucket is **public and can be accessed by anyone on the internet**. To remove public access, remove "allUsers" and "allAuthenticatedUsers" from the bucket's members.

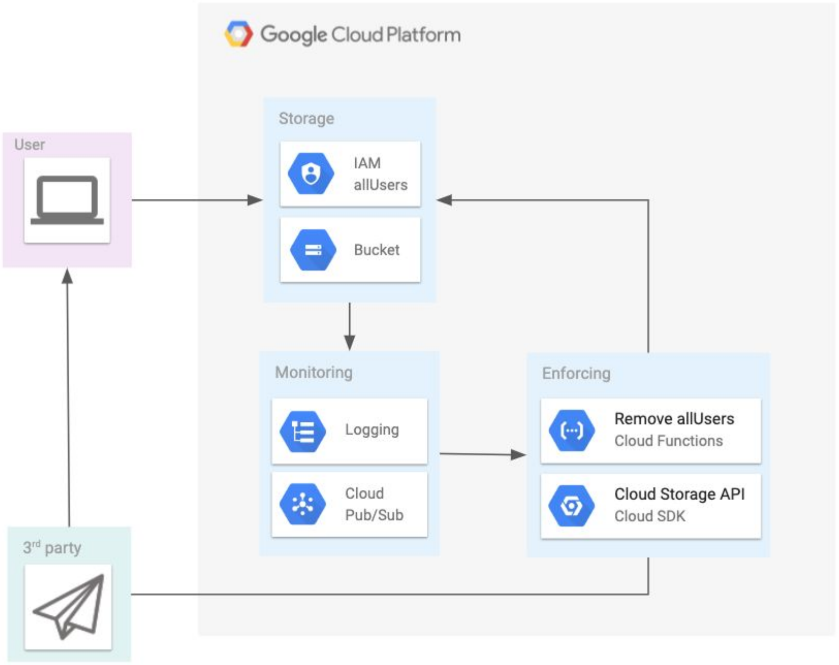
```
logName="projects/dkr-test-bsides-barcelona/logs/cloudaudit.googleapis.com%2Factivity"
protoPayload.serviceData.policyDelta.bindingDeltas.member="allUsers"
protoPayload.serviceData.policyDelta.bindingDeltas.action="ADD"
resource.type="gcs_bucket"
```

Automation with GCP Functions

- Query in Stackdriver
- Export to sink in a Pub/Sub topic
- Function listening to the topic
  - All information in the log
    - Bucket name
    - Roles added
    - Cloud Storage API

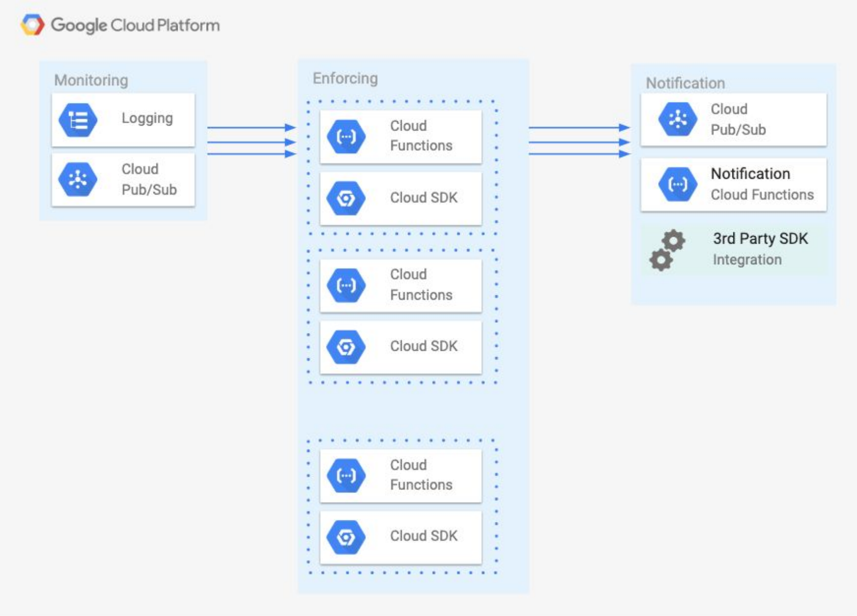


```
log = json.loads(pubsub_message)
bucket_name = log['protoPayload']['resource']['labels']['bucket_name']
bindings = log['protoPayload']['serviceData']['policyDelta']['bindingDeltas']

storage_client = storage.Client()
bucket = storage_client.bucket(bucket_name)

policy = bucket.get_iam_policy()

for binding in bindings:
    role = binding['role']
    policy[role].discard('allUsers')
    print('Role ' + role + ' removed')

bucket.set_iam_policy(policy)
```



Automation Framework

- Modules
  - Google Cloud functions with basic functionality
- Communication
  - Pub/Sub topics
- Integrations
  - Notifications
  - Tickets
- Only GCP tools in cloud environment



Conclusions

- We have all information we need to have control and visibility
- We can use the same tools in the cloud
- Useful for detection and verifying controls
- Easy to create a framework
- Good starting point for the transition from on-prem mindset to cloud

Cloud Security: Logging and Alerting

Thank you!

Questions?