Vectra XDR: Google SecOps SIEM Integration - User Guide

V 1.0.0

Contents

Overview

Vectra Platform

Vectra AI delivers an AI-driven hybrid attack detection, investigation and response platform. The Vectra AI Platform is the integrated signal powering XDR providing hybrid attack surface coverage across identity, public cloud, SaaS, and data center networks; AI-driven Attack Signal Intelligence that prioritizes real attacks in real-time and integrated, automated, and managed response services.

Google SecOps

Google SecOps is a cybersecurity telemetry platform for threat hunting, and threat intelligence and is part of the Google Cloud Platform. Google SecOps stores log events it receives in two formats: either as the original raw log or structured Unified Data Model (UDM) log. There are two critical elements to consider for parsing, Unified Data Model (UDM) which defines the schema for parsing, and Configuration Based Normalizers (CBN) which describes how log data is transformed to the UDM schema.

Looker Dashboards for Vectra XDR

This integration aims to enable seamless ingestion, parsing, and visualization of Vectra network intelligence data within Google SecOps SIEM. This integration will allow Google SecOps SIEM to receive real-time detections, hosts, accounts, health, audit, and lockdown data from Vectra using Vectra API , enriching the SIEM's threat detection and response capabilities with comprehensive network data.

Release Notes

V1.0.0

- Provided the ingestion script which deployed as a GCP cloud function to collect data from the Vectra Platform and ingest in the Google SecOps.
- Provided the parser that processes data ingested from the Vectra platform and converts it into the Google SecOps UDM data model.
- Provided below dashboards for visualization
    - Entities
    - Detection
    - Audit
    - Health
    - Lockdown
- Vectra XDR (RUX) integration, we have introduced support for log parsing under the new **VECTRA_XDR** label. Previously, logs from RUX were routed through the **VECTRA_DETECT** label, which is optimized for QUX Detect specific telemetry.

App Installation & Configuration

Pre-Requisites

- Google SecOps console.
- Vectra credentials (API Client, API Secret, Vectra Portal URL)
- GCP Project with the below required permissions:
    - GCP user and project service account should have **owner** permissions. (used to access other services within your GCP project)
- GCP Services
    - Cloud function (4-core CPU or higher is recommended for cloud function configuration)
    - GCS bucket
    - Secret Manager
    - Cloud Scheduler
- Google SecOps Service Account
    - Used to access the ingestion API for ingestion logs to the SIEM platform.
    - If you don't have the service account or are encountering this issue, you can obtain it by creating a support case with Google. [Reference]
- Looker Instance

Creating zip of the cloud function

- You can download the zip directly from the Vectra GitHub Repo from the vectra_rux_connector branch.
  OR
- Create a zip file with the contents of the following files:
  1. Download the common directory from Git repository.
  2. Download the contents of the ingestion script. (Link will be added here once it will be submitted)
  3. Create a zip in the following structure.

  ❖ Example:

```
├── README.md
├── common
│   ├── __init__.py
│   ├── auth.py
│   ├── auth_test.py
│   ├── env_constants.py
│   ├── ingest.py
│   ├── ingest_test.py
│   ├── status.py
│   ├── utils.py
│   └── utils_test.py
└── .env.yml
├── constant.py
├── exception.py
├── main.py
├── main_test.py
└── requirements.txt
└── utils.py
└── utils_test.py
└── vectra_client.py
└── vectra_client_test.py
```

Using Secrets

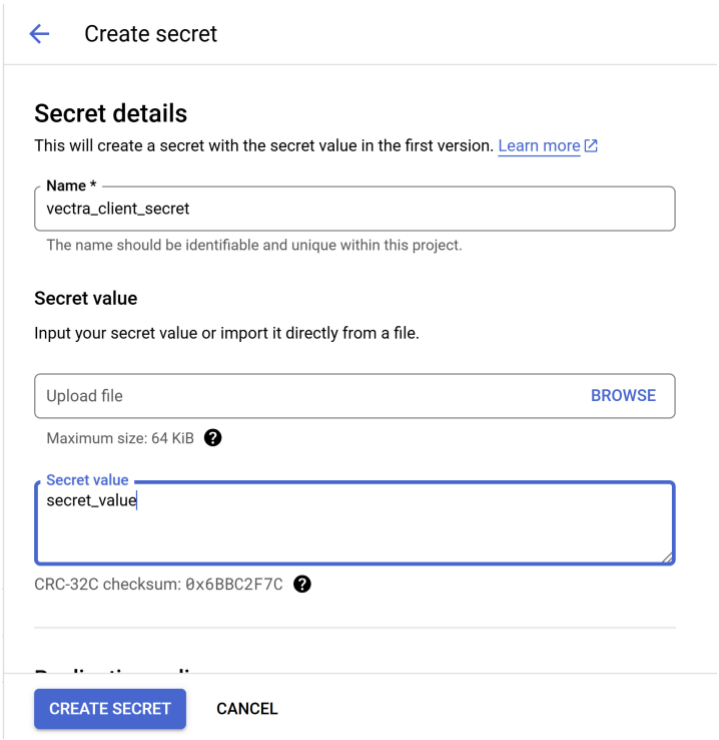
- Environment variables marked as secret must be configured as secrets on Google Secret Manager. [REF]
- Once the secrets are created on Secret Manager, use the secret's resource ID as the value for environment variables.

  For example:
  CHRONICLE_SERVICE_ACCOUNT:
  projects/{project_id}/secrets/{secret_id}/versions/{version_id}

Add the Vectra API Secret in Secret Manager

1. Log in to the "https://console.cloud.google.com/" using valid credentials.
2. Navigate to 'Secret Manager'.
3. Click on 'Create Secret'.
4. Provide the name for the secret in the 'Name' field.
5. Add your Vectra API Secret value to the Secret Value.
6. Keep the other configurations as default, Click on the 'Create Secret' button.

Similarly, Create a secret for Vectra **Client ID** and **Chronicle Service Account**.



Refer to this page for more information about how to create secrets.

Once the secrets are created on Secret Manager, use the secret's resource id as the value for environment variables. For example

SECRET_KEY: projects/{project_id}/secrets/{secret_id}/versions/{version_id}

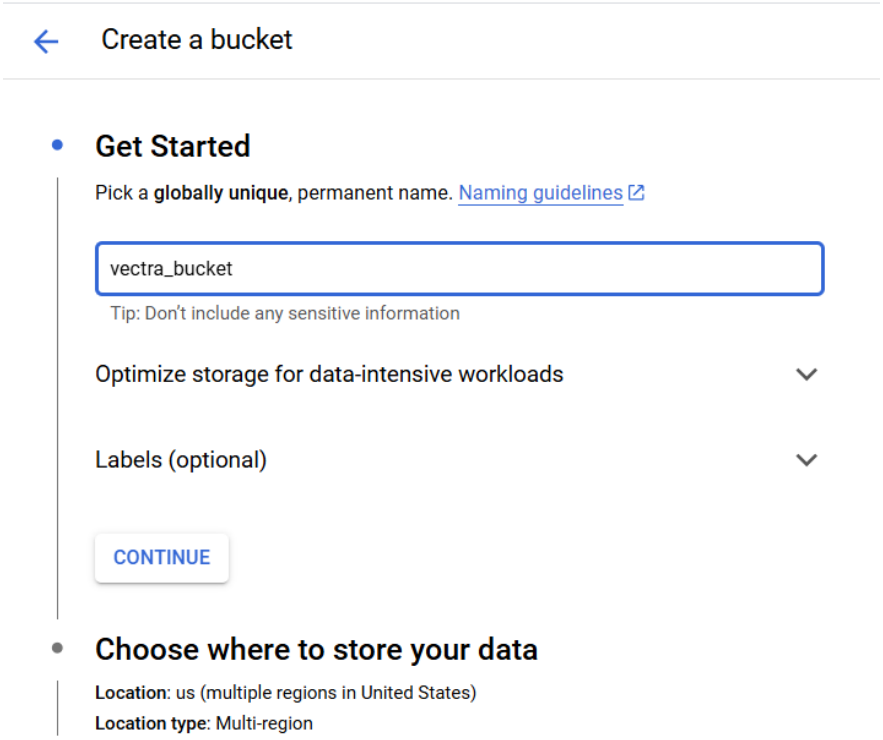
Note : Ensure that the resource ID format matches the following structure: projects/{project_id}/secrets/{secret_id}/versions/{version_id}. You can copy the resource name from the secret version details to obtain this value.

Create a GCP Bucket

---

1. Log in to the "https://console.cloud.google.com/" using valid credentials.
2. Navigate to Buckets in GCP.
3. Click on the Create button.
4. Enter the name of the bucket.
5. Users can select the region and modify the optional parameters if required and then click on the Create button.

Copy the bucket name and provide it in the GCP_BUCKET_NAME environment variable.



Cloud Function Deployment

Command based deployment

1. Navigate to the bucket and open the bucket created for the Vectra XDR in these steps. Upload the created cloud function zip file in the bucket.
2. Click Activate Cloud Shell at the top right corner of the Google Cloud console.
3. Modify the below command based on your value and run in the terminal.

Command Format :

```
gcloud functions deploy CLOUD_FUNCTION_NAME --set-env-vars
ENV_NAME1=ENV_VALUE1,ENV_NAME2=ENV_VALUE2,ENV_NAME3=ENV_VALUE
3  --gen2 --runtime=python312 --region=REGION --
source=SOURCE_OF_FUNCTION  --entry-point=main --service-
account=SERVICE_ACCOUNT_EMAIL --trigger-http --no-allow-unauthenticated --
memory=8GiB --timeout=3600s
```

- **CLOUD_FUNCTION_NAME**: Unique name of the cloud function.
- **REGION**: A region for your cloud function. (Ex : us-central1, us-west1, etc.)
- **SOURCE_OF_FUNCTION**: gsutil URI of the cloud function zip in cloud storage. (e.g. gs://vectra_xdr/function.zip) where the vectra_xdr is the name of the created bucket and function.zip is the cloud function zip file.
- **SERVICE_ACCOUNT_EMAIL**: Email of the created service account of the project. Make sure the selected Service account must have an Owner Permission. Update Service Account Permission following these steps.
- **VPC_NAME**: Name of the created VPC Network.
- **ENV_NAME1=ENV_VALUE1**: Name and value of the environment variable to be created. Environment variables

Example Command,

```
gcloud functions deploy funcusingcmd --set-env-vars
CHRONICLE_CUSTOMER_ID=abcd1234-1234-1234-abcd-
1234abcd12,CHRONICLE_SERVICE_ACCOUNT=projects/1234567890/secrets/chroni
cle_service_account/versions/1,CHRONICLE_REGION=us,GCP_BUCKET_NAME=test-
```

```
bucket,GCP_PROJECT_NUMBER=1234567890,CLIENT_ID=projects/1234567890/secrets/client_id_test,SECRET_KEY=projects/1234567890/secrets/client_secret_test,VECTRA_PORTAL_URL=https://1234567890.cc1.portal.vectra.ai,  --gen2 --runtime=python312 --region=us-central1 --source=gs://test-bucket/vectra_test.zip --entry-point=main --service-account=1234567890-compute@developer.gserviceaccount.com --trigger-http --no-allow-unauthenticated --memory=8GiB --timeout=3600s
```

Configure Scheduler

Command based deployment

1. Click Activate Cloud Shell at the top right corner of the Google Cloud console.
2. Modify the below command based on your value and run in the terminal.

Command Format :

```
gcloud scheduler jobs create http SCHEDULER_NAME --schedule="CRON_TIME" --uri="CLOUD_FUNCTION_URL" --attempt-deadline=30m --oidc-service-account-email=SERVICE_ACCOUNT_EMAIL --location=LOCATION --time-zone=TIME_ZONE
```

- **SCHEDULER_NAME**: Unique name of the cloud scheduler.
- **CRON_TIME**: Cron time format for the scheduler to run in every interval. (eg. */10 * * * *)
- **CLOUD_FUNCTION_URL**: URL of the created cloud function. Navigate to created cloud function details.
- **SERVICE_ACCOUNT_EMAIL**: Email of the created service account of the project. Make sure the selected Service account must have an Owner Permission. Update Service Account Permission following these steps.
- **LOCATION**: A region for your connector. (Ex : us-central1, us-west1, etc)
- **TIME_ZONE**: The time zone of your region. (Ex : UTC)

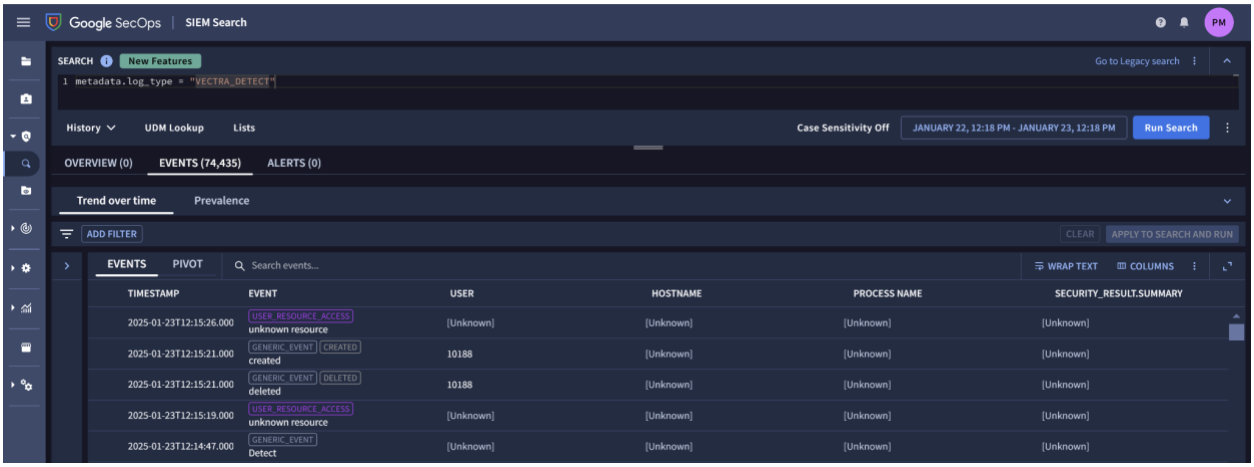
Example Command,

```
gcloud scheduler jobs create http scheduleusingcommand --schedule="*/10 * * * *" --uri=https://us-central1-alpha-000011-f5.cloudfunctions.net/testfunction --attempt-deadline=30m --oidc-service-account-email=test-alpha-000011-f5@appspot.gserviceaccount.com --location=us-central1 --time-zone=UTC
```

View Events in Google SecOps

---

1. Log in to Google SecOps:
    a. Open a web browser and navigate to the Google SecOps instance URL. For example: https://test.backstory.chronicle.security/
    b. Replace test with your actual Google SecOps instance name.
2. Access SIEM Search:
    a. From the top left corner of the Google SecOps console, select the "Investigation" option.
    b. Within the Investigation section, choose "SIEM Search".
3. Filter Events by Log Type:
    a. In the SIEM Search interface, locate the "UDM Search" section.
    b. Apply a filter for the metadata field "log_type". Set the filter value to metadata.log_type="VECTRA_DETECT".
4. View Vectra Events:
    a. The SIEM Search results will display Vectra events within the "Events" section.



Update Service Account Permission

---

1. Open **GCP Console**, Then go to **IAM**.
2. In View By **Main Tab** > Click **GRANT ACCESS**.
3. Add Service Account name in **New Principals**. (Example : service_account_name.gserviceaccount.com)
4. In **Assign Role**, Select **Owner**.
5. Click **Save**.

Environment Variables to be configured in Cloud Function

Variables	Description	Required	Datatype	Default Value	Secret
VECTRA_PORTAL_URL	Vectra portal URL	Yes	string	-	No
CLIENT_ID	Client Id generated from API Client	Yes	string	-	Yes
SECRET_KEY	Secret Key generated from API Client	Yes	string	-	Yes
CHRONICLE_CUSTOMER_ID	Google SecOps platform customer ID	Yes	string	-	No
CHRONICLE_REGION	Google SecOps platform region	No	string	us	No
CHRONICLE_SERVICE_ACCOUNT	Contents of the SecOps service account JSON file	Yes	string	-	Yes
GCP_BUCKET_NAME	GCP bucket name.	Yes	string	-	No
GCP_PROJECT_NUMBER	GCP project number where secrets are created.	Yes	string	-	No
INCLUDE_TRIAGED	Whether to include Detection events that have been triaged.	No	bool	false	No
INCLUDE_SCORE_DECREASES	Whether to include a score decreases the entity events.	No	bool	false	No

INCLUDE_INFO_CATEGORY	Whether to include Detection events with info category.	No	bool	true	No
VLANS	Configure v_lans for Health data.	No	bool	false	No
HISTORICAL	If set true, collect 24Hr Historical data.	No	bool	false	No
ENABLE_SCORING	Enables the ingestion and processing of scoring events.	No	bool	true	No
ENABLE_DETECTION	Enables the ingestion and processing of detection events.	No	bool	true	No
ENABLE_HEALTH	Enables the ingestion and processing of health events.	No	bool	true	No
ENABLE_AUDIT	Enables the ingestion and processing of audit events.	No	bool	true	No
ENABLE_LOCKDOWN	Enables the ingestion and processing of lockdown events.	No	bool	true	No

Looker Installation & Configuration

Pre-Requisites

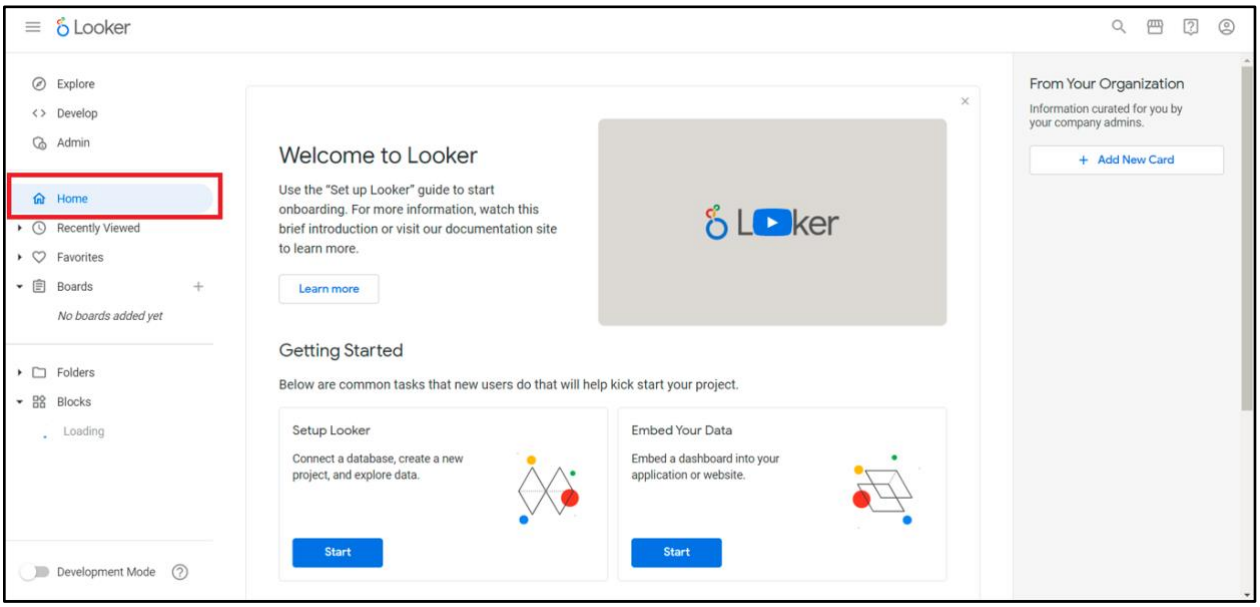
- Billing Project ID, Dataset name, and Service account file of BigQuery that stores Google SecOps data for database connection in Looker.
- BigQuery Export feature needs to be enabled for your Google SecOps tenant. (Reach out to your Google SecOps representative to set this up.)

User Permissions

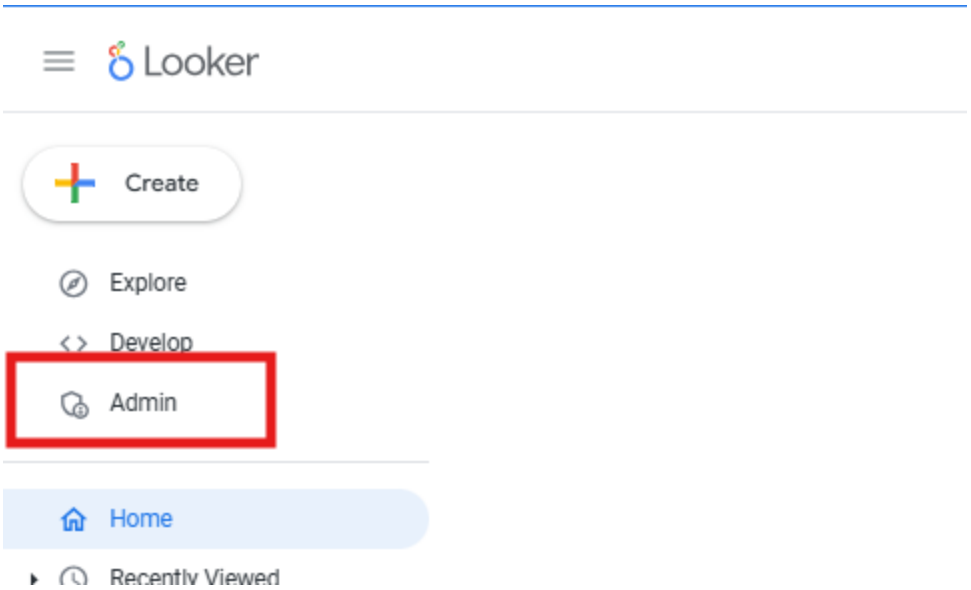
- Admin Role User - to create database connections and install blocks from the marketplace.

Create a connection to Google SecOps in Looker

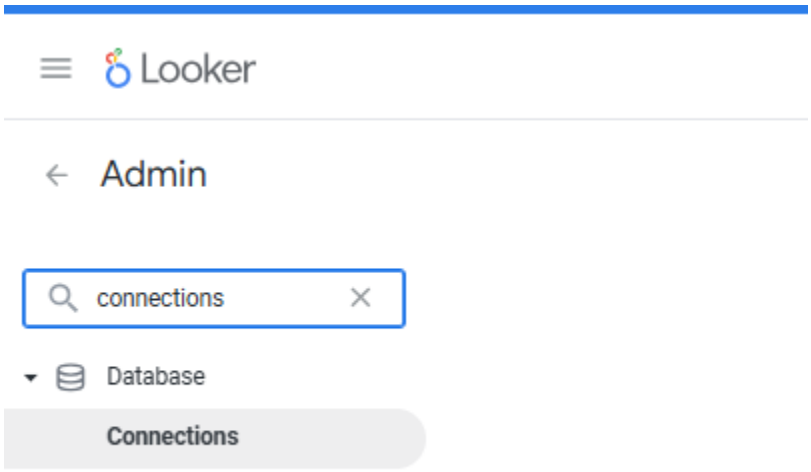
1. To create a connection to Google SecOps, first open the Looker instance and navigate to the Home page.



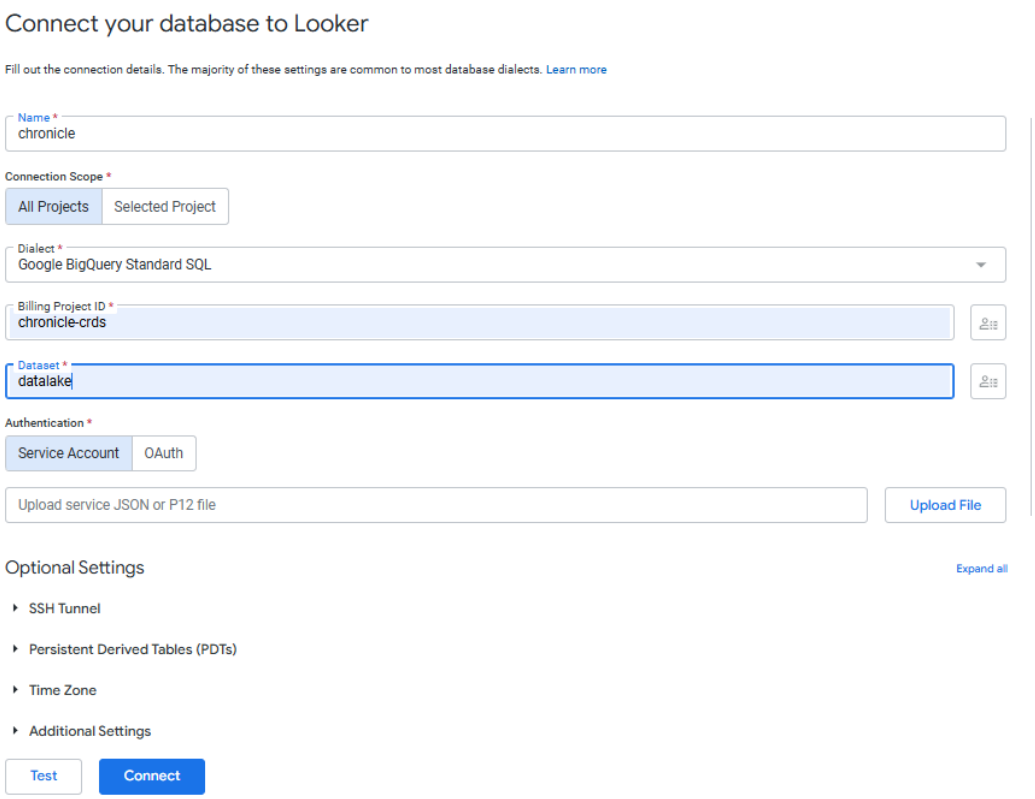
2. Now click on the **Admin** from the main menu (in left panel)



3. Now type **Connections** in the search, once the Connections option appears click on it to see the connection page.



4. Now click on the **Add connection**() to create a new connection and name it as **chronicle**.
5. Select **Google BigQuery Standard SQL** in the Dialect. Now several new fields will appear.
6. Enter Billing Project ID field. Example: "**chronicle-crds**" here, where Chronicle data is present.
7. Enter the **datalake** in the Dataset name.



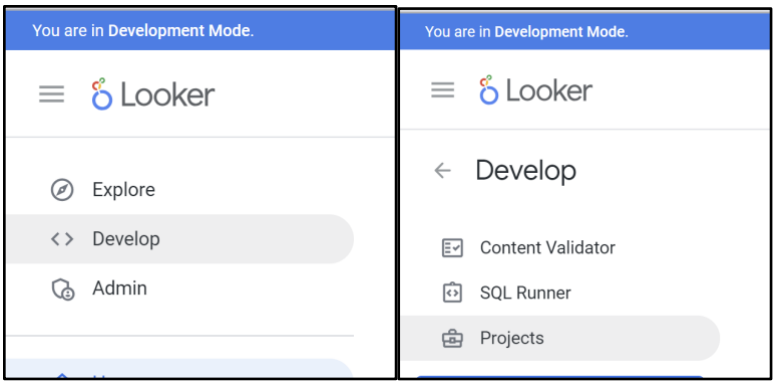
8. To configure authentication, select the service account method and upload your Chronicle service account file.
9. In the optional settings, set both the timestamps (Database timestamp and query timestamp) as UTC (the time fields shown in dashboards will be populated accordingly).
10. Click on Test to check the connectivity of Looker with Google Chronicle database.
11. Click on the Connect button (

) to complete the connection setup. Looker is now connected to the Google Chronicle database.

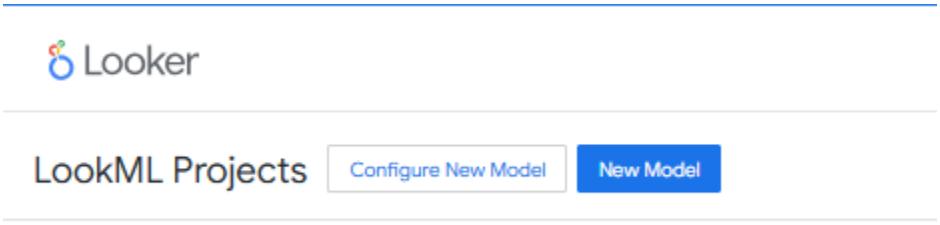
Get the Block from GitHub Repository

1. Go to Vectra looker dashboard github repository and fork the same. Make sure to uncheck the option for fork only the vectra_rux_dashboard branch.
2. Go to Looker and turn on “Development Mode” from the sidebar panel.

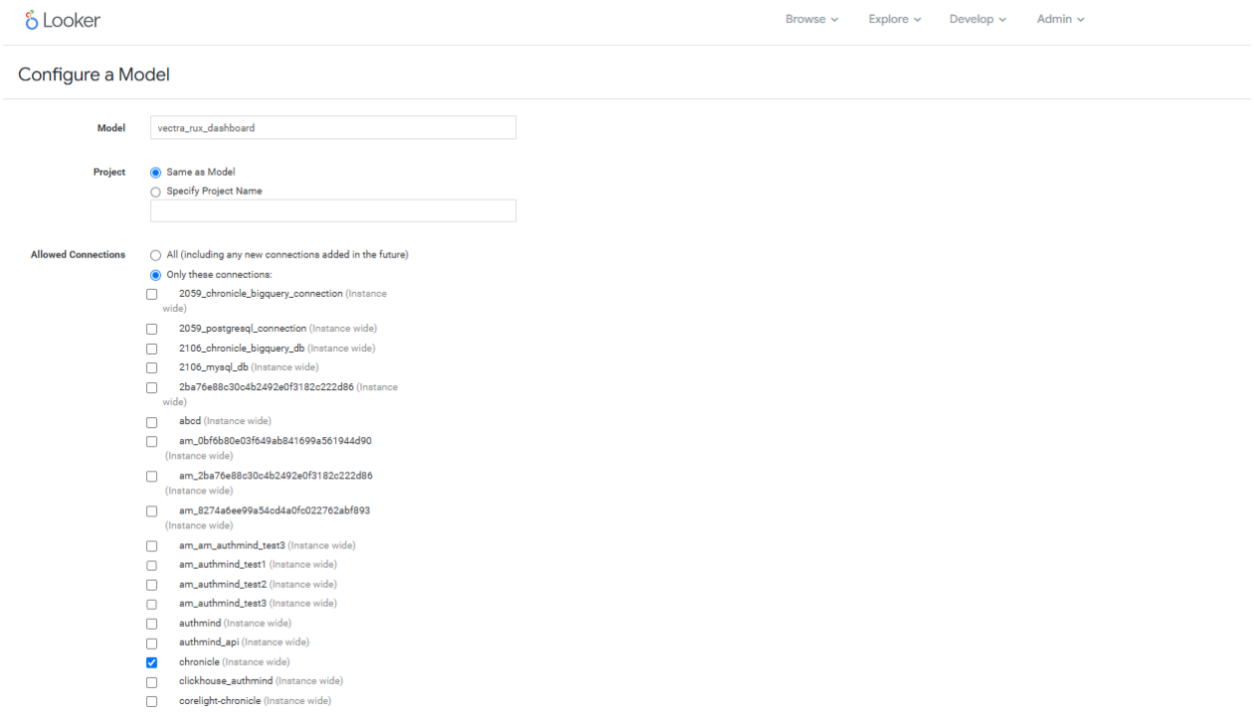
3. Select Projects from the Develop menu.



4. From the LookML Projects page, select **Configure New Model** to open the model configuration page.

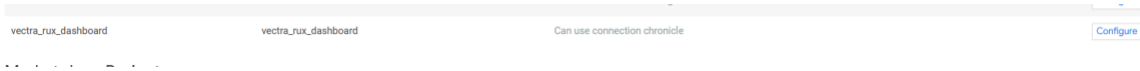


5. Enter the model name as **vectra_rux_dashboard** and keep the **Same as Model** option selected.

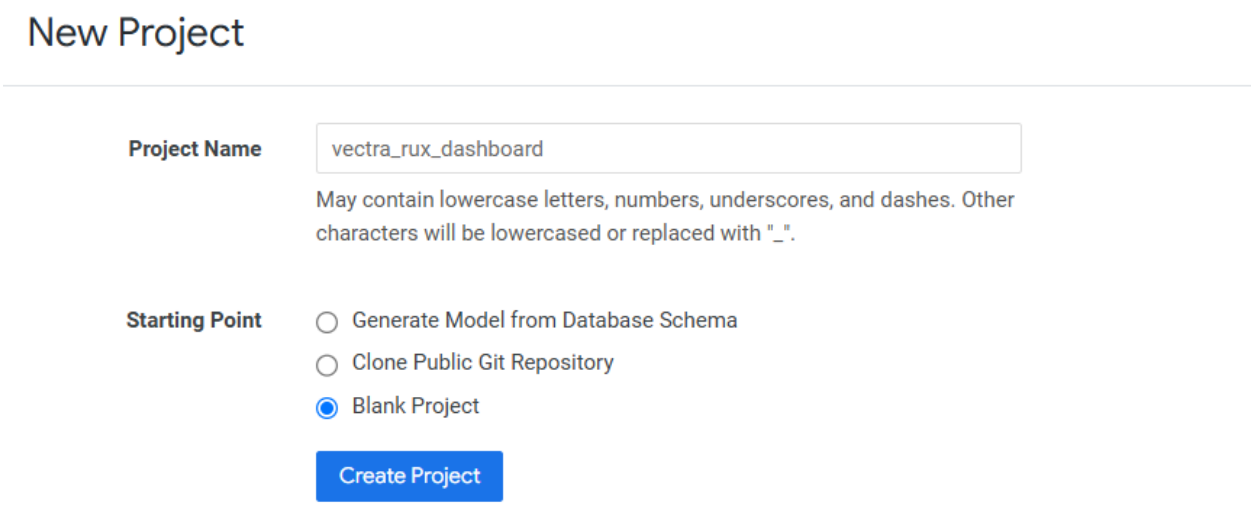


6. Click on **Save** to save the configuration.

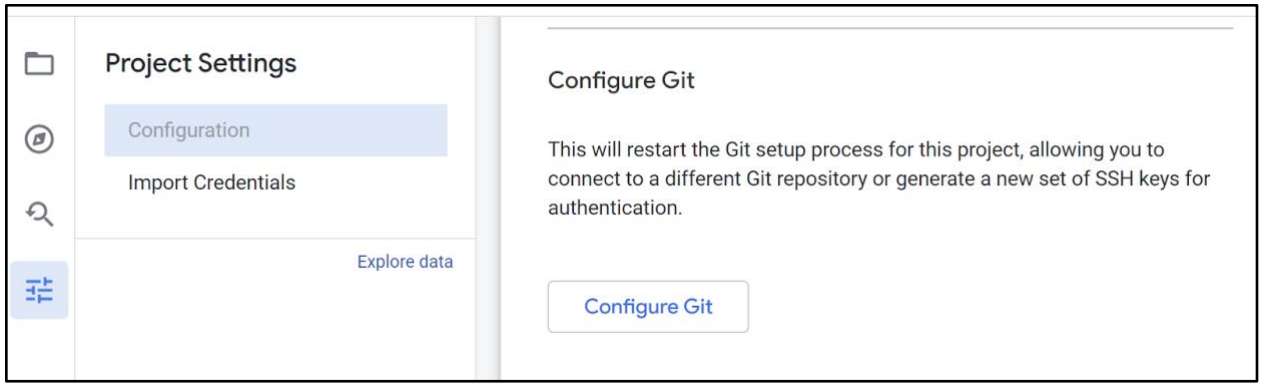
7. After saving the configuration you'll be redirected to the **Projects** page and you'll find your project in the **Pending Project** section.
8. Click on **Add LookML** to configure your project and add the lookML file in it.



9. On the New Project page, configure these options for your new project:
   Project Name: Give project name **'vectra_rux_dashboard'**.
   Starting Point: Select **Blank Project**.
   Click on **Create Project**. The project will be created and opened in the Looker IDE.

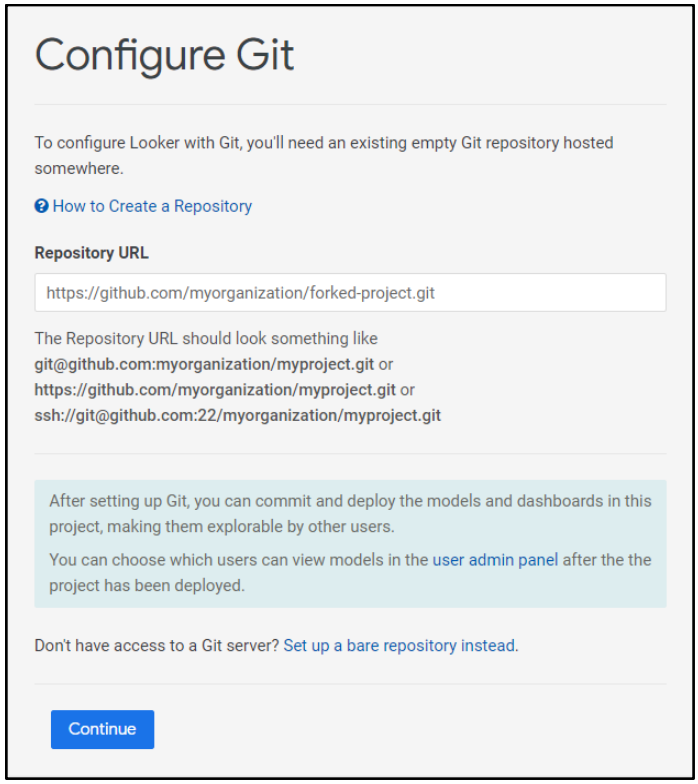


10. Click on the Settings icon from the navigation bar, and open the Configure Git page by selecting the Configure Git button.



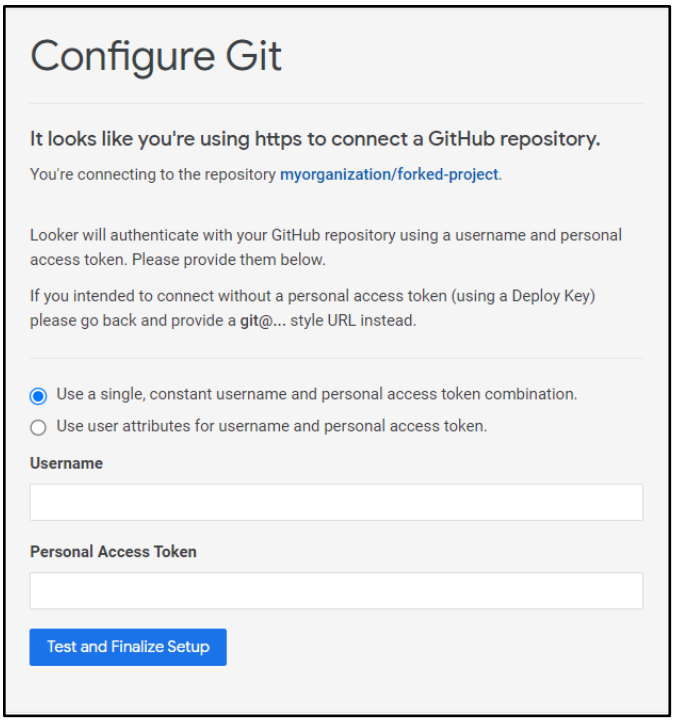
11. In Looker's Configure Git section, paste the URL of the forked Vectra Looker Dashboard Git Repository in the Repository URL field, then select Continue.

    e.g. https://github.com/<your_username>/looker-dashboards.git

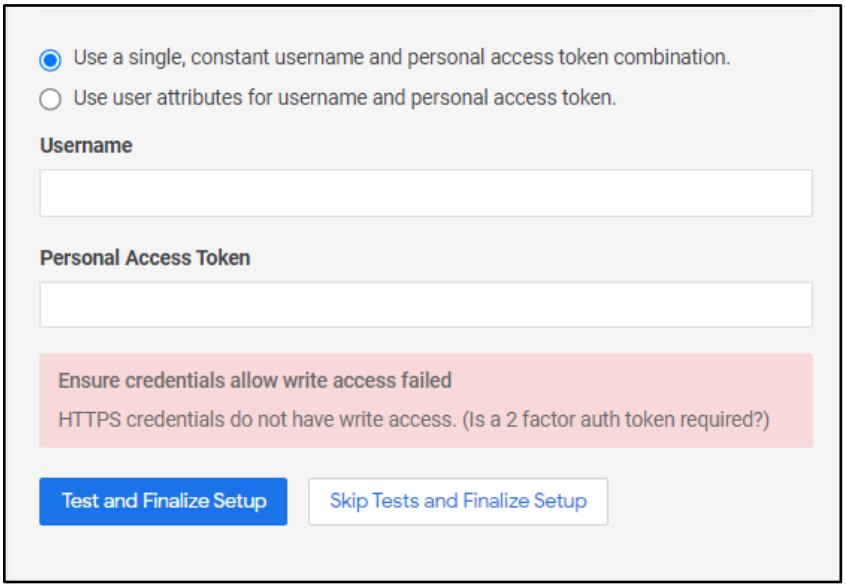


12. Enter the github username and Personal Access Token, then click “Test and Finalize Setup”.

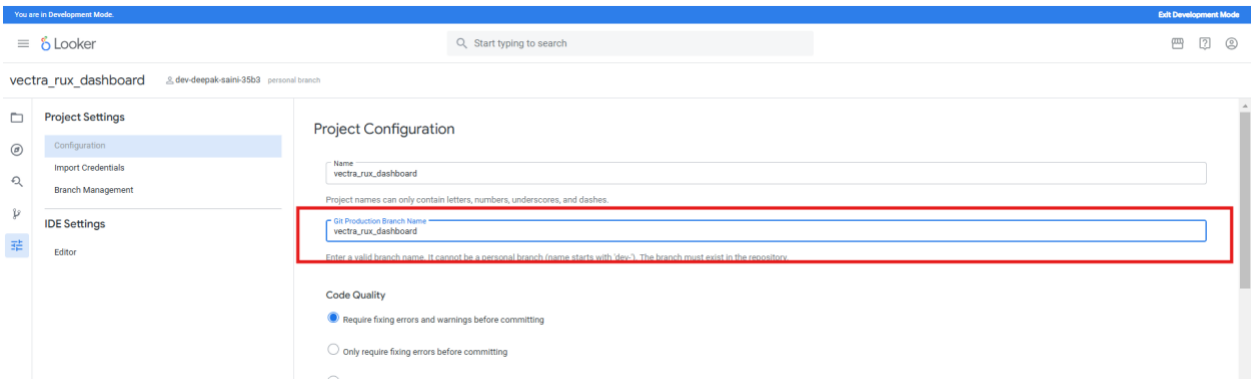
    Note: Make sure the **Personal Access Token (PAT)** you created from your github repository has **Read/Write** permissions of the repository.



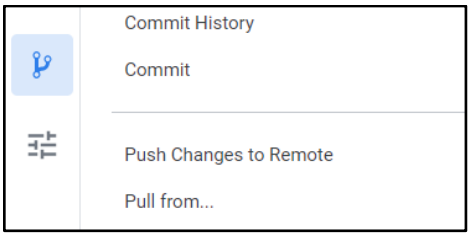
13. If you get an error like “Ensure credential allow write access failed”, just enter the username and token again and click “Skip Tests and Finalize Setup”.



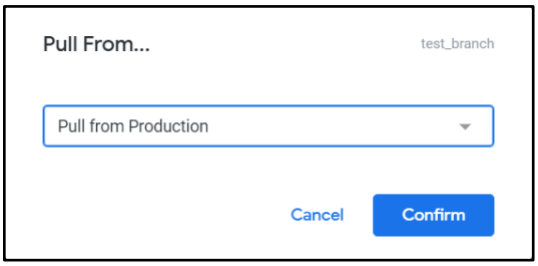
14. Once the git is configured, open the project settings and change the production branch to **vectra_rux_dashboard**



15. Now, you should be able to see the code in your project from the **vectra_rux_dashboard** branch. If not then,
    a. In the 'Git Actions' tab from the left side, click on the "Pull from..." option.

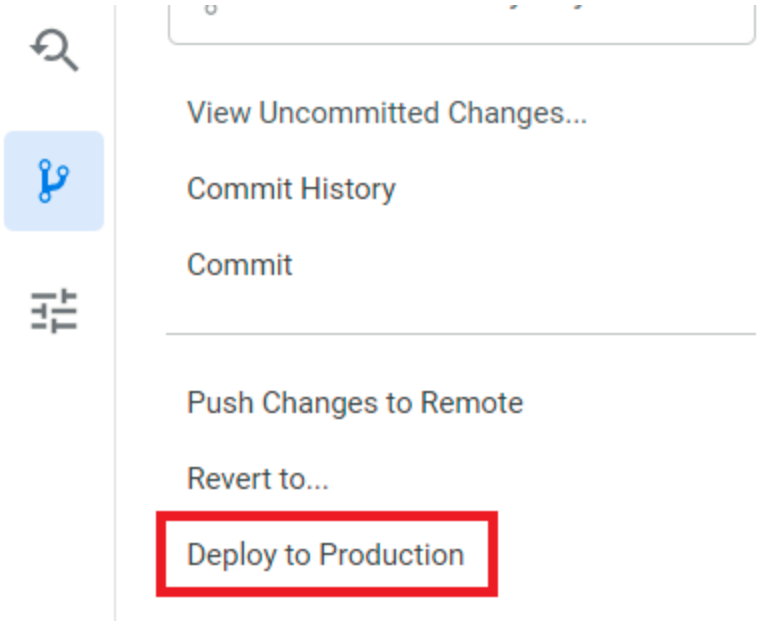


    b. Select the "Pull From Production" option and click on the Confirm button.

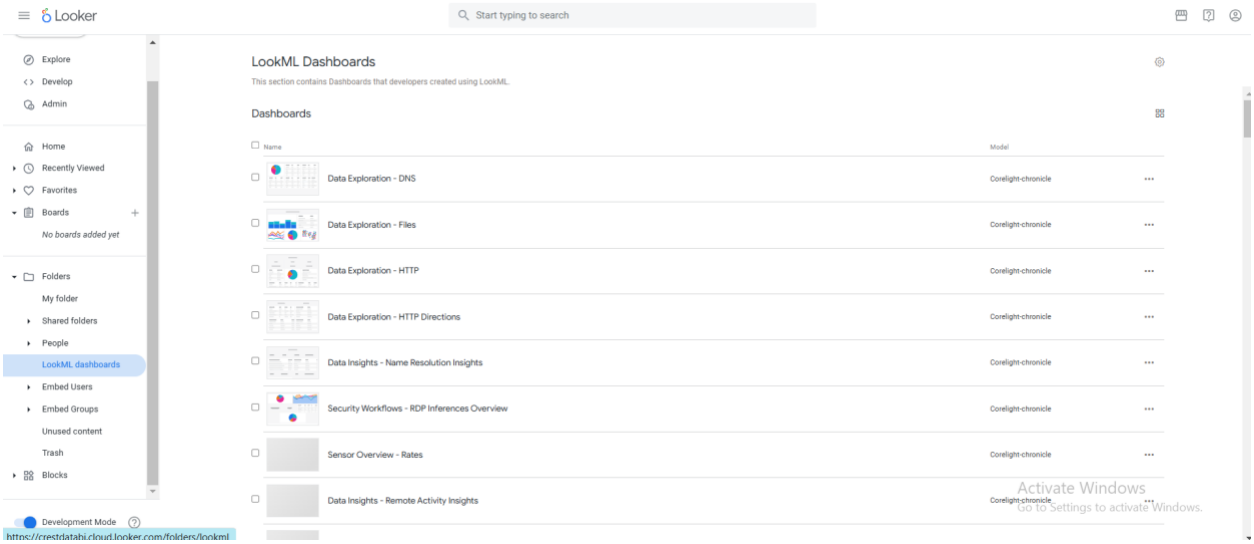


16. Any changes can be committed and pushed to the dev branch by clicking on **Validate LookML** and once the files are validated click on **Commit and Push changes**.
17. After the above steps, In the Git Actions, click on the "Deploy to Production" or you can press "**Deploy to Production**" from the top right corner.

    Note: 'Deploy to Production' will push code to the production branch that is set in the project settings. By default, it will be the vectra_rux_dashboard branch. If you don't want to push code to the 'main' branch, then create your own branch and set it to 'Git Production Branch Name' in project settings. Then click on Deploy to Production.



18. Now you can turn off the development mode in order to see the LookML dashboards.
19. On the Homepage of your Looker instance, navigate to the "**LookML dashboards**" tab under the "**Folders**" tab to access and view all the dashboards.



20. The connection name defined at the top of the **vectra_rux_dashboard.model** file must match the connection name created earlier. If the user has named it chronicle, no changes are necessary. Otherwise, the connection field needs to be updated with the correct connection name.

Dashboards

Entities Dashboard

This dashboard displays data for the "Scoring" log type, including incidents that are either prioritized or not. The table panel presents details about each incident, including their most recent update status.

1. Filters description as per label

Log type

- This filter contains a single value for the dashboard, and other filters will update based on the selection made in the **Log Type** filter. **Default**: Scoring.

Timerange

- This filter updated the panel based on the time range selected in it. **Default:** Last 7 days.

Entity Type

- Filters the Panels according to the selected entity type **i.e.** account or host. **Default**: all.

Prioritized

- Filters the Panels according to the selected priority **i.e.** true or false. **Default**: all.

Data Source Type

- Filters the Panels according to the selected data source type. Filter have the following values AWS, O365, M365, SAML and Network **Default**: all.

2. Panel description as per label

Prioritized

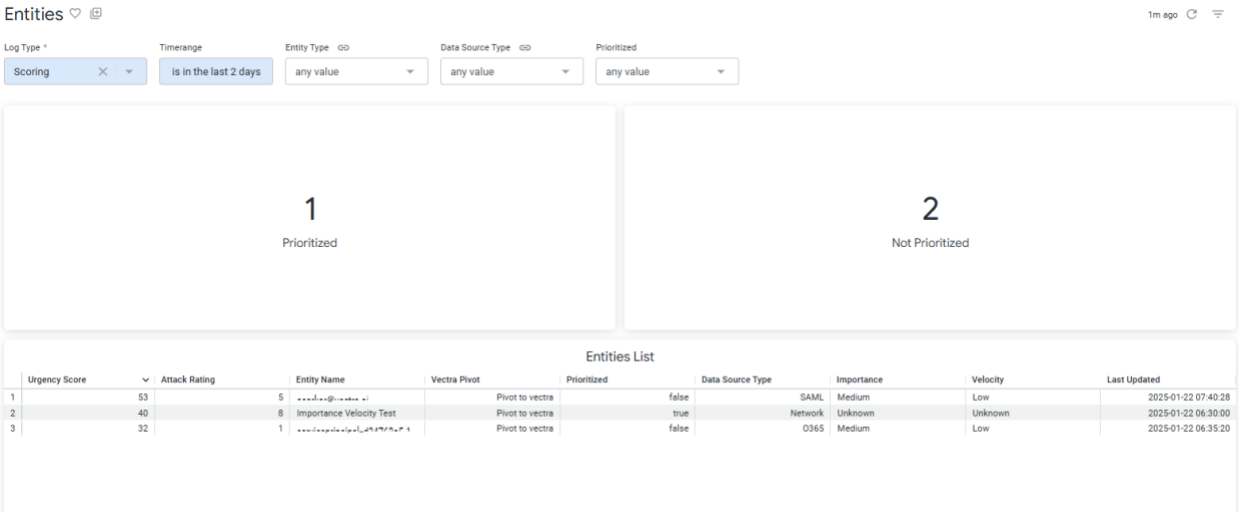
- The single value panel displays the count of prioritized incidents based on their most recent updates, which were collected from the Vectra RUX platform sent to Google SecOps.

Not Prioritized

- The single value panel displays the count of unprioritized incidents based on their most recent updates, which were collected from the Vectra RUX platform sent to Google SecOps.

Entities List

- The table panel displays the incidents of prioritized and unprioritized incidents based on their most recent updates, which were collected from the Vectra RUX platform sent to Google SecOps.



Detection Dashboard

1. Filters description as per label

Log type

- This filter contains a single value for the dashboard, and other filters will update based on the selection made in the **Log Type** filter. **Default**: Detection.

Timerange

- This filter updated the panel based on the time range selected in it. **Default:** Last 7 days.

Data Source Type

- Filters the Panels according to the selected data source type. Filter have the following values AWS, O365, M365, SAML and Network **Default**: all.

Behaviour

- Filters the Panels according to the selected behaviour. **Default**: all.

Category

- Filters the Panels according to the selected category. **Default**: all.

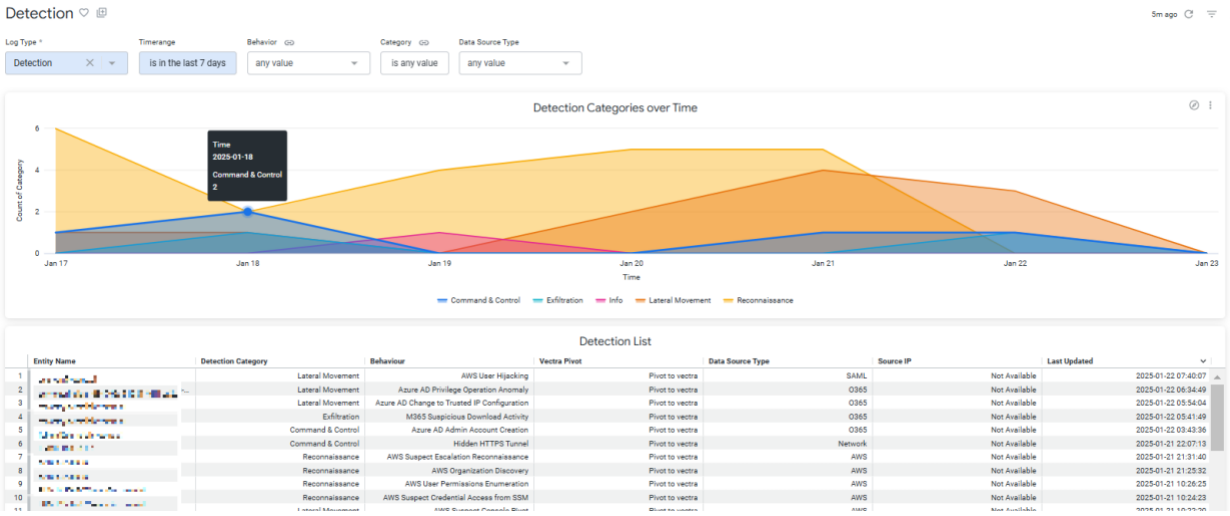
2. Panel description as per label

Detection Categories over Time

- This displays an area chart for the count of the categories over time of Detection log.

Detection List

- The table panel displays the incidents of detection logs based on their most recent updates, which were collected from the Vectra RUX platform sent to Google SecOps.



Audit Dashboard

1. Filters description as per label

Log type

- This filter contains a single value for the dashboard, and other filters will update based on the selection made in the **Log Type** filter. **Default**: Audit.

Timerange

- This filter updated the panel based on the time range selected in it. **Default:** Last 7 days.

Username

- Filters the Panels according to the selected username. **Default**: all.

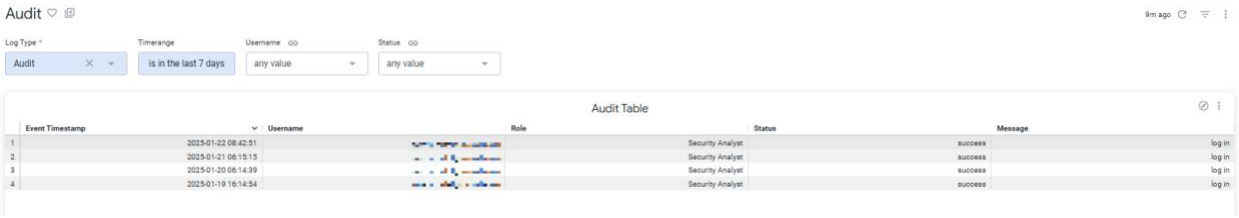
Status

- Filters the Panels according to the selected status. **Default**: all.

2. Panel description as per label

Audit Table

- The table panel displays the incidents of audit logs based on their most recent updates, which were collected from the Vectra RUX platform sent to Google SecOps.



Lockdown Dashboard

1. Filters description as per label

Log type

- This filter contains a single value for the dashboard, and other filters will update based on the selection made in the **Log Type** filter. **Default**: Lockdown.

Timerange

- This filter updated the panel based on the time range selected in it. **Default:** Last 7 days.

Entity Type

- Filters the Panels according to the selected entity type **i.e.** account or host. **Default**: all.

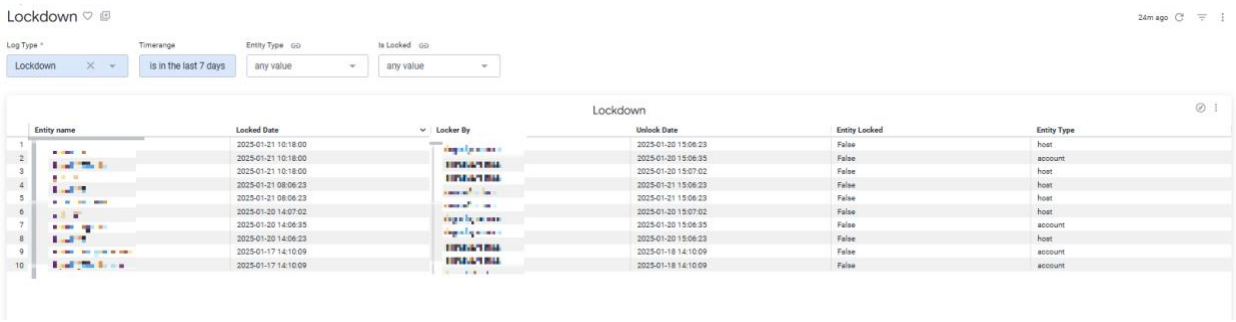
Is Locked

- Filters the Panels according to the selected “Is locked” **i.e.** true or false. **Default**: all.

2. Panel description as per label

Lockdown

- The table panel displays the incidents of lockdown logs based on their most recent updates, which were collected from the Vectra XDR platform sent to Google SecOps. It shows whether the incident is still locked or not.



Health Dashboard

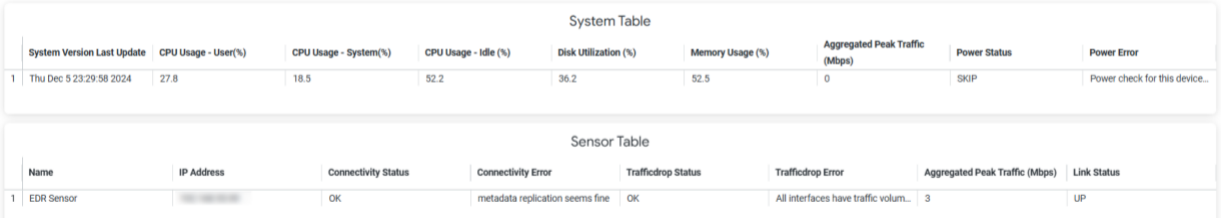
1. Panel description as per label

System Table

- This panel displays the most recent data of the system's resource utilization.

Sensors Table

- This panel displays the most recent data related to sensors like connectivity status, connectivity error, trafficdrop status, trafficdrop error etc.



Notes

- If an optional environment variable is not provided during the Cloud Function deployment, default values will be used, and data collection will start accordingly.
- The chunk limit for data collection is set to 100 to minimize data duplication in case of errors during ingestion, as the Google SecOps Ingestion API processes data in chunks of 100.
- We recommend setting the timeout in the RUNTIME variable to the maximum value (3600) to prevent the Cloud Function from terminating during data collection.
- The dashboard displays data in a tabular format and will show only the top 100 incidents, sorted by the selected field in the table.
- The Data Source Type will no longer include static values like Network, Microsoft 365 / Azure AD, and AWS. Instead, it will feature AWS, M365, O365, SAML, and Network, and filter data accordingly.

Limitations

- If the user does not specify the required environment variable while configuring the Cloud Function, the script deployment will fail.
- CBN parser will only be able to parse the Vectra RUX events.
- We suggest using the second generation of Cloud Function. The first generation of Cloud Function has a maximum execution time of 9 minutes and the second generation of Cloud Function has a maximum execution time of 60 minutes. If the execution time of the Cloud Function exceeds timeout then there are chances that the complete data won't be ingested in the Google SecOps.
- The rate limit for a Vectra account depends on the user's subscription. Based on this API rate limit, the integration will be able to collect data and ingest into Google SecOps. Once the API rate limit is exceeded, data collection will only resume when the limit is reset after a specific interval.
- Looker doesn't support API calls to fetch live data for populating dashboards.
- We do not have any marketplace for Google SecOps integration, so as a part of deliverables, we use to push our integration code to google public repo which undergoes many checks from google side.
- Common search filters can't be implemented in looker.
- Pagination in tabular visualization is not supported in Looker.
- Looker will only show data from the past 180 days, but this can vary as per the retention policy configured in BigQuery.
- According to the query time zone selected by the user in connection with the Google SecOps database, the Looker dashboards would be reflected according to the configured timezone.
- The looker dashboard does not display data in the drill down table when there are too many records to be displayed.

Troubleshooting

This section describes the common issues that might happen during the deployment or the running of the app and the steps to resolve the issues.

1. GCloud logs can be used for troubleshooting.
    a. Log in to the "https://console.cloud.google.com/" using valid credentials.
    b. Navigate to 'Cloud functions' and click on the deployed function where you can find the logs module.
    c. Logs can be filtered using severity.

    Currently, this logs feature is disabled by the google team in some GCP projects. We are currently checking with the google team regarding this.
2. If you test the cloud function immediately after deploying it on gcloud, It might be possible that the cloud function will not work as expected. To resolve this, wait for a few seconds and then test it.
3. If the cloud function stops its execution because memory exceeds the limit, reconfigure the cloud function's memory configuration and increase the memory limit.
4. The Looker dashboard takes data from the cache and does not display the latest events.

    Solution:

    1. Click on the three dots present on the rightmost side of the dashboard.
    2. Click on the Clear cache and refresh.

5. The data is not displayed on the dashboard -

This could be a problem with the data source as the database connection might be wrongly configured.

6. If desired events are not showing in the visualization -
   Make sure that the filters in the dashboard are configured correctly. If the filters are too restrictive, they may be preventing the dashboard from displaying any data.
7. The dashboard may be slow to load or unresponsive - This could be due to a problem with the data source being unavailable or having too much data, the query that is being used, or the way that the dashboard is being rendered.

GCP Resources/Services Approximate Cost Details

Service	Standard Configurations	Purpose	Reference
Cloud Functions	Type: Memory: 8192MB CPU: 4.8GHz Execution time per function (ms): 3600 Invocations per month: 1500 Minimum number of instances: 1	Function / Script which pulls data from Vectra Platform using API and ingest into Google SecOps.	Approx cost ~ $66/month https://cloud.google.com/functions/pricing
Cloud Storage	Total Amount of Storage: 1 GiB	Storage bucket used to manage API checkpoints	Approx cost ~ $0.02/month https://cloud.google.com/storage/pricing
Secret Manager	Access operations: 1500	Used to maintain credentials.	Approx cost ~ $0/month https://cloud.google.com/secret-manager/pricing
Cloud Scheduler	Total amount of jobs: 1	Scheduler which executes above cloud function at a specific time interval.	Approx cost ~ $0/month https://cloud.google.com/scheduler/pricing
Looker	https://cloud.google.com/looker/pricing#platform_editions	Visualization tool to visualize events from Google SecOps.	https://cloud.google.com/looker/pricing Cost depends on the edition which we select from the link on the left column.

Note: Users can also calculate (using pricing calculator) the estimated price of the Google Cloud services used.

References

- Install the gcloud CLI
- Deploying cloud functions from local machine
- Looker
- Looker Marketplace