Postgres and Github

GitHub Personal Access Token (PAT)

To authenticate with the GitHub MCP server, you need a GitHub Personal Access Token.

1. Go to your GitHub Developer settings.
2. Click on "Personal access tokens" -> "Tokens (classic)".
3. Click "Generate new token" -> "Generate new token (classic)".
4. Give your token a descriptive name.
5. Set an expiration date for your token.
6. Important: For security, grant your token the most limited scopes necessary. For read-only access to repositories, the `repo:status`, `public_repo`, and `read:user` scopes are often sufficient. Avoid granting full repo or admin permissions unless absolutely necessary.
7. Click "Generate token".
8. Copy the generated token.

Vertex AI (PAT)

To use Vertex AI, you will need to create a Google Cloud project and enable Vertex AI.

Authenticate and enable Vertex AI API:

```
gcloud auth login
# Replace <your_project_id> with your project ID
gcloud config set project <your_project_id>
gcloud services enable aiplatform.googleapis.com
```

Create a `.env` file by running the following (replace `<your_project_id>` with your project ID and `<your_github_pat_here>` with your GitHub Personal Access Token):

```
echo "GOOGLE_GENAI_USE_VERTEXAI=TRUE" >> .env \
&& echo "GOOGLE_CLOUD_PROJECT=<your_project_id>" >> .env \
&& echo "GOOGLE_CLOUD_LOCATION=us-central1" >> .env \
&& echo "GITHUB_PERSONAL_ACCESS_TOKEN=<your_github_pat_here>" >> .env
```

There is an example `.env` file located at .env.example if you would like to verify your `.env` was set up correctly.

Source the `.env` file into your environment:

```
set -o allexport && source .env && set +o allexport
```

MCP toolbox

Download [MCP Toolbox for Databases]

```
export OS="linux/amd64" # one of linux/amd64, darwin/arm64, darwin/amd64, or windows/amd64
curl -O --output-dir deployment/mcp-toolbox https://storage.googleapis.com/genai-toolbox/v0.6.0/$OS/toolbox
chmod +x deployment/mcp-toolbox/toolbox
```

Before you begin

Deploying to Google Cloud requires:

- A Google Cloud project with billing enabled.
- `gcloud` CLI (Installation instructions)

1 - Authenticate the Google Cloud CLI, and enable Google Cloud APIs.

```
gcloud auth login
gcloud auth application-default login

export PROJECT_ID="<YOUR_PROJECT_ID>"
gcloud config set project $PROJECT_ID

gcloud services enable sqladmin.googleapis.com \
   compute.googleapis.com \
   cloudresourcemanager.googleapis.com \
   servicenetworking.googleapis.com \
   aiplatform.googleapis.com
```

2 - Create a Cloud SQL (Postgres) instance.

```
gcloud sql instances create software-assistant \
   --database-version=POSTGRES_16 \
```

```
    --tier=db-custom-1-3840 \
    --region=us-central1 \
    --edition=ENTERPRISE \
    --enable-google-ml-integration \
    --database-flags cloudsql.enable_google_ml_integration=on \
    --root-password=admin
```

Once created, you can view your instance in the Cloud Console here.

3 - Create a SQL database, and grant Cloud SQL service account access to Vertex AI.

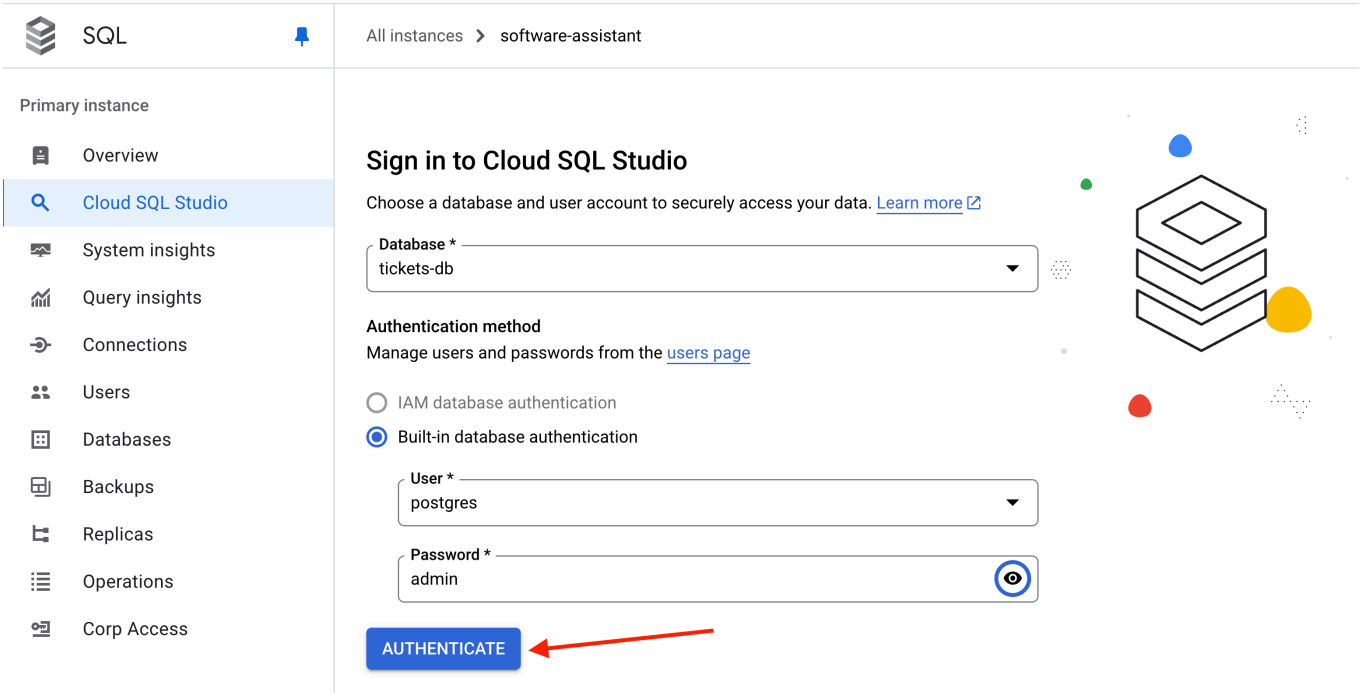
This step is necessary for creating vector embeddings (Agent RAG search).

```
gcloud sql databases create tickets-db --instance=software-assistant

SERVICE_ACCOUNT_EMAIL=$(gcloud sql instances describe software-assistant --
format="value(serviceAccountEmailAddress)")
echo $SERVICE_ACCOUNT_EMAIL

gcloud projects add-iam-policy-binding $PROJECT_ID --
member="serviceAccount:$SERVICE_ACCOUNT_EMAIL" --role="roles/aiplatform.user"
```

4 - Set up the `tickets` table.

From the Cloud Console (Cloud SQL), open **Cloud SQL Studio**.

Log into the `tickets-db` Database using the `postgres` user (password: `admin`, but note you can change to a more secure password under Cloud SQL > Primary Instance > Users).



Open a new **Editor** tab. Then, paste in the following SQL code to set up the table and create vector embeddings.

```
CREATE EXTENSION IF NOT EXISTS google_ml_integration CASCADE;
CREATE EXTENSION IF NOT EXISTS vector CASCADE;
GRANT EXECUTE ON FUNCTION embedding TO postgres;

CREATE TABLE tickets (
    ticket_id SERIAL PRIMARY KEY,             -- PostgreSQL's auto-
incrementing integer type (SERIAL is equivalent to INT AUTO_INCREMENT)
    title VARCHAR(255) NOT NULL,              -- A concise summary or title of
the bug/issue.
    description TEXT,                         -- A detailed description of the
bug.
    assignee VARCHAR(100),                    -- The name or email of the
person/team assigned to the ticket.
    priority VARCHAR(50),                     -- The priority level (e.g., 'P0
- Critical', 'P1 - High').
    status VARCHAR(50) DEFAULT 'Open',        -- The current status of the
ticket (e.g., 'Open', 'In Progress', 'Resolved'). Default is 'Open'.
    creation_time TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP, --
Timestamp when the ticket was first created. 'WITH TIME ZONE' is recommended
for clarity and compatibility.
    updated_time TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP  --
Timestamp when the ticket was last updated. Will be managed by a trigger.
);
```

5 - Load in sample data.

From Cloud SQL Studio, paste in the following SQL code to load in sample data.

```
INSERT INTO tickets (title, description, assignee, priority, status) VALUES
('Login Page Freezes After Multiple Failed Attempts', 'Users are reporting
that after 3 failed login attempts, the login page becomes unresponsive and
requires a refresh. No specific error message is displayed.',
'samuel.green@example.com', 'P0 - Critical', 'Open');

INSERT INTO tickets (title, description, assignee, priority, status) VALUES
('Dashboard Sales Widget Intermittent Data Loading Failure', 'The "Sales
Overview" widget on the main dashboard intermittently shows a loading spinner
but no data. Primarily affects Chrome browser users.',
'maria.rodriguez@example.com', 'P1 - High', 'In Progress');

INSERT INTO tickets (title, description, assignee, priority, status) VALUES
('Broken Link in Footer - Privacy Policy', 'The "Privacy Policy" hyperlink
located in the website footer leads to a 404 "Page Not Found" error.',
'maria.rodriguez@example.com', 'P3 - Low', 'Resolved');

INSERT INTO tickets (title, description, assignee, priority, status) VALUES
('UI Misalignment on Mobile Landscape View (iOS)', 'On specific iOS devices
(e.g., iPhone 14 models), the top navigation bar shifts downwards when the
device is viewed in landscape orientation, obscuring content.',
'maria.rodriguez@example.com', 'P2 - Medium', 'In Progress');

INSERT INTO tickets (title, description, assignee, priority, status) VALUES
('Critical XZ Utils Backdoor Detected in Core Dependency (CVE-2024-3094)',
'Urgent: A sophisticated supply chain compromise (CVE-2024-3094) has been
identified in XZ Utils versions 5.6.0 and 5.6.1. This malicious code
potentially allows unauthorized remote SSH access by modifying liblzma.
Immediate investigation and action required for affected Linux/Unix systems
and services relying on XZ Utils.', 'frank.white@example.com', 'P0 -
Critical', 'Open');

INSERT INTO tickets (title, description, assignee, priority, status) VALUES
('Database Connection Timeouts During Peak Usage', 'The application is
experiencing frequent database connection timeouts, particularly during peak
hours (10 AM - 12 PM EDT), affecting all users and causing service
interruptions.', 'frank.white@example.com', 'P1 - High', 'Open');

INSERT INTO tickets (title, description, assignee, priority, status) VALUES
('Export to PDF Truncates Long Text Fields in Reports', 'When generating PDF
exports of reports containing extensive text fields, the text is abruptly cut
off at the end of the page instead of wrapping or continuing to the next
```

```
page.', 'samuel.green@example.com', 'P1 - High', 'Open');

INSERT INTO tickets (title, description, assignee, priority, status) VALUES
('Search Filter "Date Range" Not Applying Correctly', 'The "Date Range" filter
on the search results page does not filter records accurately; results outside
the specified date range are still displayed.', 'samuel.green@example.com',
'P2 - Medium', 'Resolved');

INSERT INTO tickets (title, description, assignee, priority, status) VALUES
('Typo in Error Message: "Unathorized Access"', 'The error message displayed
when a user attempts an unauthorized action reads "Unathorized Access" instead
of "Unauthorized Access."', 'maria.rodriguez@example.com', 'P3 - Low',
'Resolved');

INSERT INTO tickets (title, description, assignee, priority, status) VALUES
('Intermittent File Upload Failures for Large Files', 'Users are
intermittently reporting that file uploads fail without a clear error message
or explanation, especially for files exceeding 10MB in size.',
'frank.white@example.com', 'P1 - High', 'Open');
```

6 - Create a trigger to update the `updated_time` field when a record is updated.

```
CREATE OR REPLACE FUNCTION update_updated_time_tickets()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_time = NOW();  -- Set the updated_time to the current
timestamp
    RETURN NEW;                -- Return the new row
END;
$$ language 'plpgsql';

CREATE TRIGGER update_tickets_updated_time
BEFORE UPDATE ON tickets
FOR EACH ROW                  -- This means the trigger fires for each row
affected by the UPDATE statement
EXECUTE PROCEDURE update_updated_time_tickets();
```

7 - Create vector embeddings from the `description` field.

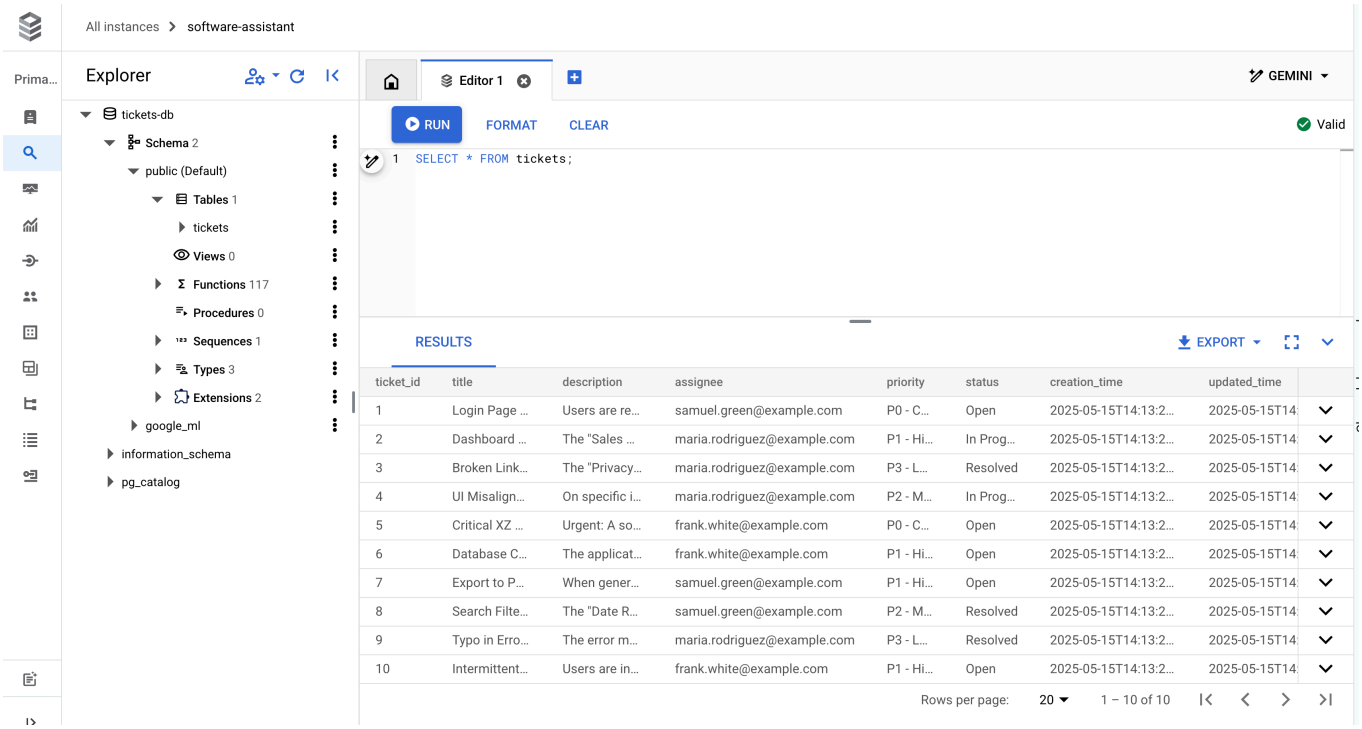
```
ALTER TABLE tickets ADD COLUMN embedding vector(768) GENERATED ALWAYS AS
(embedding('text-embedding-005',description)) STORED;
```

8 - Verify that the database is ready.

From Cloud SQL studio, run:

```
SELECT * FROM tickets;
```

You should see:



9 - Deploy the MCP Toolbox for Databases server to Cloud Run

Now that we have a Cloud SQL database, we can deploy the MCP Toolbox for Databases server to Cloud Run and point it at our Cloud SQL instance.

First, update `deployment/mcp-toolbox/tools.yaml` for your Cloud SQL instance:

```yaml
sources:
  postgresql: # GCP -  CLOUD SQL
    kind: cloud-sql-postgres
    project: linear-theater-463712-r8
    region: us-central1
    instance: software-assistant
    database: tickets-db
    user: postgres
    password: admin
```

Then, configure Toolbox's Cloud Run service account to access both Secret Manager and Cloud SQL. Secret Manager is where we'll store our `tools.yaml` file because it contains sensitive Cloud SQL credentials.

Note - run this from the top-level `software-bug-assistant/` directory.

```
gcloud services enable run.googleapis.com \
   cloudbuild.googleapis.com \
   artifactregistry.googleapis.com \
   iam.googleapis.com \
   secretmanager.googleapis.com

gcloud iam service-accounts create toolbox-identity

gcloud projects add-iam-policy-binding $PROJECT_ID \
    --member serviceAccount:toolbox-
identity@$PROJECT_ID.iam.gserviceaccount.com \
    --role roles/secretmanager.secretAccessor

gcloud projects add-iam-policy-binding $PROJECT_ID \
    --member serviceAccount:toolbox-
identity@$PROJECT_ID.iam.gserviceaccount.com \
    --role roles/cloudsql.client

gcloud secrets create tools --data-file=deployment/mcp-toolbox/tools.yaml
```

Now we can deploy Toolbox to Cloud Run. We'll use the latest release version of the MCP Toolbox image (we don't need to build or deploy the `toolbox` from source.)

```
gcloud run deploy toolbox --image us-central1-docker.pkg.dev/database-
toolbox/toolbox/toolbox:latest --service-account toolbox-identity@linear-
theater-463712-r8.iam.gserviceaccount.com --region us-central1 --set-
secrets="/app/tools.yaml=tools:latest" --set-env-vars="PROJECT_ID=linear-
theater-463712-r8,DB_USER=postgres,DB_PASS=admin"  --args="--tools-
file=/app/tools.yaml" --args="--address=0.0.0.0" --args="--port=8080" --allow-
unauthenticated --memory=1Gi --cpu=1 --timeout=300 --port=8080
```

Verify that the Toolbox is running by getting the Cloud Run logs:

```
gcloud run services logs read toolbox --region us-central1
```

You should see:

```
2025-05-15 18:03:55 2025-05-15T18:03:55.465847801Z INFO "Initialized 1
sources."
2025-05-15 18:03:55 2025-05-15T18:03:55.466152914Z INFO "Initialized 0
authServices."
2025-05-15 18:03:55 2025-05-15T18:03:55.466374245Z INFO "Initialized 9 tools."
2025-05-15 18:03:55 2025-05-15T18:03:55.466477938Z INFO "Initialized 2
toolsets."
2025-05-15 18:03:55 2025-05-15T18:03:55.467492303Z INFO "Server ready to
serve!"
```

Save the Cloud Run URL for the Toolbox service as an environment variable.

```
export MCP_TOOLBOX_URL=$(gcloud run services describe toolbox --region us-
central1 --format "value(status.url)")
```

Now we are ready to deploy the ADK Python agent to Cloud Run! 🚀

10 - Create an Artifact Registry repository.

This is where we'll store the agent container image.

```
gcloud artifacts repositories create adk-samples \
  --repository-format=docker \
  --location=us-central1 \
  --description="Repository for ADK Python sample agents" \
  --project=$PROJECT_ID
```

11 - Containerize the ADK Python agent.

Build the container image and push it to Artifact Registry with Cloud Build.

```
gcloud builds submit --region=us-central1 --tag us-central1-
docker.pkg.dev/$PROJECT_ID/adk-samples/software-bug-assistant:latest
```

12 - Deploy the agent to Cloud Run

```
gcloud run deploy software-bug-assistant \
    --image=us-central1-docker.pkg.dev/linear-theater-463712-r8/adk-
samples/software-bug-assistant:latest \
    --region=us-central1 \
    --allow-unauthenticated \
    --set-env-vars="GOOGLE_CLOUD_PROJECT=linear-theater-463712-
```

```
r8,GOOGLE_CLOUD_LOCATION=us-
central1,GOOGLE_GENAI_USE_VERTEXAI=TRUE,MCP_TOOLBOX_URL=https://toolbox-
yts7e77tuq-
uc.a.run.app,GITHUB_PERSONAL_ACCESS_TOKEN=ghp_7DnskXcmact8y6Cs82A3WDS4MEtE3C3c
3P0c"
```

When this runs successfully, you should see:

```
Service [software-bug-assistant] revision [software-bug-assistant-00001-d4s]
has been deployed and is serving 100 percent of traffic.
```

13 - Test the Cloud Run Agent

Open the Cloud Run Service URL outputted by the previous step.

You should see the ADK Web UI for the Software Bug Assistant.

Test the agent by asking questions like:

- `Any issues around database timeouts?`
- `How many bugs are assigned to samuel.green@example.com? Show a table.`
- `What are some possible root-causes for the unresponsive login page issue?` (Invoke Google Search tool)
- `Get the bug ID for the unresponsive login page issues` --> `Boost that bug's priority to P0.`.
- `Create a new bug.` (let the agent guide you through bug creation)