I'll update both README files with the corrected architecture diagrams. Here are the updated versions:

MCP Toolbox for BigQuery and Cloud Databases

A comprehensive implementation of the Model Context Protocol (MCP) Toolbox for integrating Google BigQuery and other cloud databases with Claude Desktop and Google AI Development Kit (ADK). This repository demonstrates how to build database-powered AI agents using natural language interfaces.

Table of Contents

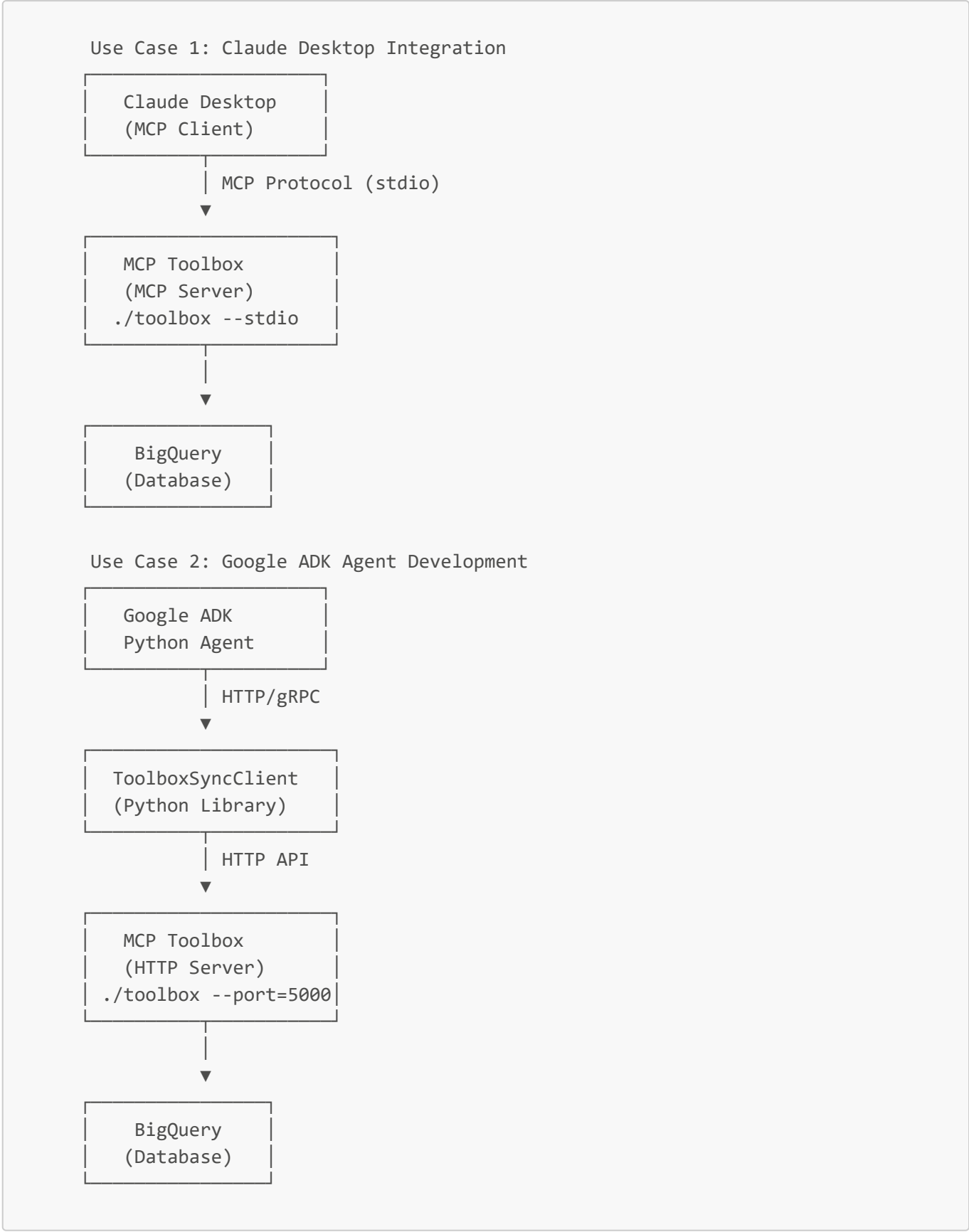
Overview

The MCP Toolbox enables AI models to interact directly with databases through a standardized protocol. This implementation focuses on Google BigQuery integration, providing:

- **Natural Language Database Queries**: Convert user questions into SQL queries automatically
- **Multi-Database Support**: Connect to BigQuery, Cloud SQL, AlloyDB, Spanner, and 20+ other databases
- **Secure Authentication**: Service account and Application Default Credentials support
- **Agent Development**: Build custom AI agents using Google's ADK framework
- **Claude Desktop Integration**: Direct database access from Claude's desktop application

Architecture

The MCP Toolbox supports two distinct use cases with separate architectures:



Key Points:

- Claude Desktop and Google ADK are completely separate, independent systems
- Both can use MCP Toolbox but through different interfaces:
    - Claude Desktop: Uses MCP protocol via stdio communication

    - Google ADK: Uses HTTP/gRPC via the ToolboxSyncClient Python library
- There is no direct connection between Claude Desktop and Google ADK

Prerequisites

System Requirements

- **Operating System**: Windows 10+, Ubuntu 20.04+, macOS 11+
- **Memory**: Minimum 4GB RAM (8GB recommended)
- **Disk Space**: 500MB for toolbox and dependencies
- **Network**: Internet connection for cloud database access

Software Dependencies

1. **Google Cloud SDK** (required for BigQuery access)
2. **Claude Desktop** (for MCP client integration)
3. **Python 3.8+** (for ADK agent development)
4. **Git** (for repository management)

Google Cloud Requirements

- Active Google Cloud Project
- BigQuery API enabled
- Service account with appropriate permissions
- Billing enabled for production use

Installation

Windows Installation

1. Install Google Cloud SDK

```
# Download the installer
Invoke-WebRequest -Uri
https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe -
OutFile GoogleCloudSDKInstaller.exe

# Run the installer (follow GUI prompts)
.\GoogleCloudSDKInstaller.exe

# Initialize gcloud
gcloud init
```

2. Download MCP Toolbox Binary

```
# Set version
$VERSION = "0.15.0"
```

```
# Download Windows binary
Invoke-WebRequest -Uri "https://storage.googleapis.com/genai-
toolbox/v$VERSION/windows/amd64/toolbox.exe" -OutFile toolbox.exe

# Verify installation
.\toolbox.exe --version
```

3. Configure Environment Variables

```
# Set Google Cloud credentials
$env:GOOGLE_APPLICATION_CREDENTIALS = "D:\repos\mcp-toolbox\complete-tube-421007-
208a4862c992.json"

# Add to system environment variables (persistent)
[System.Environment]::SetEnvironmentVariable('GOOGLE_APPLICATION_CREDENTIALS',
'D:\repos\mcp-toolbox\complete-tube-421007-208a4862c992.json',
[System.EnvironmentVariableTarget]::User)
```

Linux Installation

1. Install Google Cloud SDK

```
# Add the Cloud SDK distribution URI as a package source
echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg]
https://packages.cloud.google.com/apt cloud-sdk main" | sudo tee -a
/etc/apt/sources.list.d/google-cloud-sdk.list

# Import the Google Cloud public key
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key --
keyring /usr/share/keyrings/cloud.google.gpg add -

# Update and install the Cloud SDK
sudo apt-get update && sudo apt-get install google-cloud-cli

# Additional components
sudo apt-get install google-cloud-cli-app-engine-python
sudo apt-get install google-cloud-cli-app-engine-python-extras

# Initialize gcloud
gcloud init
```

2. Download MCP Toolbox Binary

```
# Set version
export VERSION=0.15.0
```

```
# Download Linux binary
curl -O https://storage.googleapis.com/genai-toolbox/v$VERSION/linux/amd64/toolbox
chmod +x toolbox

# Move to system path (optional)
sudo mv toolbox /usr/local/bin/

# Verify installation
toolbox --version
```

3. Configure Environment Variables

```
# Add to ~/.bashrc or ~/.zshrc
export GOOGLE_APPLICATION_CREDENTIALS="/path/to/your/service-account-key.json"

# Apply changes
source ~/.bashrc
```

macOS Installation

```
# Install via Homebrew (recommended)
brew tap googleapis/toolbox
brew install mcp-toolbox

# Or download binary directly
export VERSION=0.15.0

# For Apple Silicon
curl -O https://storage.googleapis.com/genai-
toolbox/v$VERSION/darwin/arm64/toolbox

# For Intel Macs
curl -O https://storage.googleapis.com/genai-
toolbox/v$VERSION/darwin/amd64/toolbox

chmod +x toolbox
```

Google Cloud Setup

Service Account Configuration

1. Create Service Account

```
# Set your project ID
export PROJECT_ID="complete-tube-421007"
```

```
# Create service account
gcloud iam service-accounts create mcp-toolbox-sa \
    --display-name="MCP Toolbox Service Account" \
    --project=$PROJECT_ID

# Grant BigQuery permissions
gcloud projects add-iam-policy-binding $PROJECT_ID \
    --member="serviceAccount:mcp-toolbox-sa@$PROJECT_ID.iam.gserviceaccount.com" \
    --role="roles/bigquery.user"

gcloud projects add-iam-policy-binding $PROJECT_ID \
    --member="serviceAccount:mcp-toolbox-sa@$PROJECT_ID.iam.gserviceaccount.com" \
    --role="roles/bigquery.dataEditor"
```

2. Generate Service Account Key

```
# Create and download key
gcloud iam service-accounts keys create service-account-key.json \
    --iam-account=mcp-toolbox-sa@$PROJECT_ID.iam.gserviceaccount.com

# Set environment variable
export GOOGLE_APPLICATION_CREDENTIALS="$(pwd)/service-account-key.json"
```

BigQuery Dataset Setup

1. Create Dataset and Table

```
-- Create dataset
CREATE SCHEMA IF NOT EXISTS `complete-tube-421007.test`
OPTIONS(
  location="US",
  description="Hotel booking system for MCP Toolbox demo"
);

-- Create hotels table
CREATE OR REPLACE TABLE `complete-tube-421007.test.hotels` (
  id INT64 NOT NULL,
  name STRING NOT NULL,
  location STRING NOT NULL,
  price_per_night NUMERIC(10,2),
  available_rooms INT64,
  rating FLOAT64,
  booked BOOL DEFAULT FALSE,
  checkin_date DATE,
  checkout_date DATE,
  amenities ARRAY<STRING>,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP(),
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP()
```

```
);

-- Insert sample data
INSERT INTO `complete-tube-421007.test.hotels`
(id, name, location, price_per_night, available_rooms, rating, amenities)
VALUES
  (1, 'Grand Plaza Hotel', 'New York', 250.00, 50, 4.5, ['WiFi', 'Pool', 'Gym']),
  (2, 'Seaside Resort', 'Miami Beach', 180.00, 30, 4.2, ['Beach Access', 'Spa',
'Restaurant']),
  (3, 'Mountain Lodge', 'Aspen', 350.00, 20, 4.8, ['Ski Access', 'Fireplace', 'Hot
Tub']),
  (4, 'City Center Inn', 'Chicago', 120.00, 40, 3.9, ['WiFi', 'Parking',
'Breakfast']),
  (5, 'Luxury Suites', 'Las Vegas', 400.00, 25, 4.7, ['Casino', 'Pool',
'Entertainment']);
```

Configuration

Tool Definition Files

toolsdb.yaml - Hotel Booking System

```
sources:
  my-bigquery-source:
    kind: bigquery
    project: complete-tube-421007
    location: US  # Match your dataset location

tools:
  search-hotels-by-name:
    kind: bigquery-sql
    source: my-bigquery-source
    description: Search for hotels by name with partial matching
    parameters:
      - name: name
        type: string
        description: Hotel name or partial name
        required: true
    statement: |
      SELECT
        id,
        name,
        location,
        price_per_night,
        available_rooms,
        rating,
        booked,
        ARRAY_TO_STRING(amenities, ', ') as amenities_list
      FROM `test.hotels`
      WHERE LOWER(name) LIKE LOWER(CONCAT('%', @name, '%'))
      ORDER BY rating DESC;
```

```yaml
  search-hotels-by-location:
    kind: bigquery-sql
    source: my-bigquery-source
    description: Find hotels in a specific location
    parameters:
      - name: location
        type: string
        description: City or area name
        required: true
    statement: |
      SELECT
        id,
        name,
        location,
        price_per_night,
        available_rooms,
        rating,
        booked
      FROM `test.hotels`
      WHERE LOWER(location) LIKE LOWER(CONCAT('%', @location, '%'))
      ORDER BY price_per_night ASC;

  search-available-hotels:
    kind: bigquery-sql
    source: my-bigquery-source
    description: Get all available (not booked) hotels
    statement: |
      SELECT
        id,
        name,
        location,
        price_per_night,
        available_rooms,
        rating
      FROM `test.hotels`
      WHERE booked = FALSE
      ORDER BY rating DESC
      LIMIT 10;

  book-hotel:
    kind: bigquery-sql
    source: my-bigquery-source
    description: Book a hotel room by ID
    parameters:
      - name: hotel_id
        type: integer
        description: Unique hotel identifier
        required: true
      - name: checkin_date
        type: string
        description: Check-in date (YYYY-MM-DD)
        required: true
      - name: checkout_date
```

```
        type: string
        description: Check-out date (YYYY-MM-DD)
        required: true
    statement: |
      UPDATE `test.hotels`
      SET
        booked = TRUE,
        checkin_date = PARSE_DATE('%Y-%m-%d', @checkin_date),
        checkout_date = PARSE_DATE('%Y-%m-%d', @checkout_date),
        updated_at = CURRENT_TIMESTAMP()
      WHERE id = @hotel_id AND booked = FALSE;

  cancel-booking:
    kind: bigquery-sql
    source: my-bigquery-source
    description: Cancel a hotel booking
    parameters:
      - name: hotel_id
        type: integer
        description: Hotel ID to cancel
        required: true
    statement: |
      UPDATE `test.hotels`
      SET
        booked = FALSE,
        checkin_date = NULL,
        checkout_date = NULL,
        updated_at = CURRENT_TIMESTAMP()
      WHERE id = @hotel_id;

  get-booking-details:
    kind: bigquery-sql
    source: my-bigquery-source
    description: Get current booking information
    parameters:
      - name: hotel_id
        type: integer
        description: Hotel ID
        required: true
    statement: |
      SELECT
        name,
        location,
        price_per_night,
        booked,
        FORMAT_DATE('%Y-%m-%d', checkin_date) as checkin,
        FORMAT_DATE('%Y-%m-%d', checkout_date) as checkout,
        DATE_DIFF(checkout_date, checkin_date, DAY) as nights,
        price_per_night * DATE_DIFF(checkout_date, checkin_date, DAY) as
total_cost
      FROM `test.hotels`
      WHERE id = @hotel_id;

toolsets:
```

```
  hotel-management:
    - search-hotels-by-name
    - search-hotels-by-location
    - search-available-hotels
    - book-hotel
    - cancel-booking
    - get-booking-details
```

tools.yaml - Release Notes System

```
sources:
  gcp-public-data:
    kind: bigquery
    project: complete-tube-421007

tools:
  search_release_notes_recent:
    kind: bigquery-sql
    source: gcp-public-data
    description: Get recent Google Cloud release notes from the last N days
    parameters:
      - name: days_back
        type: integer
        description: Number of days to look back
        default: 7
    statement: |
      SELECT
        product_name,
        description,
        published_at,
        release_note_type,
        product_version
      FROM
        `bigquery-public-data.google_cloud_release_notes.release_notes`
      WHERE
        DATE(published_at) >= DATE_SUB(CURRENT_DATE(), INTERVAL @days_back DAY)
      ORDER BY published_at DESC
      LIMIT 50;

  search_release_notes_by_product:
    kind: bigquery-sql
    source: gcp-public-data
    description: Search release notes for a specific GCP product
    parameters:
      - name: product_name
        type: string
        description: GCP product name (e.g., 'BigQuery', 'Cloud Storage')
        required: true
    statement: |
      SELECT
        product_name,
```

```
        description,
        published_at,
        release_note_type,
        product_version
      FROM
        `bigquery-public-data.google_cloud_release_notes.release_notes`
      WHERE
        LOWER(product_name) LIKE LOWER(CONCAT('%', @product_name, '%'))
      ORDER BY published_at DESC
      LIMIT 20;

toolsets:
  release_notes_tools:
    - search_release_notes_recent
    - search_release_notes_by_product
```

Environment Variables

Create a `.env` file (add to `.gitignore`):

```
# Google Cloud Configuration
GOOGLE_APPLICATION_CREDENTIALS=/path/to/service-account-key.json
GOOGLE_CLOUD_PROJECT=complete-tube-421007
BIGQUERY_DATASET=test
BIGQUERY_LOCATION=US

# MCP Toolbox Configuration
TOOLBOX_PORT=5000
TOOLBOX_HOST=127.0.0.1
TOOLBOX_LOG_LEVEL=info

# Optional: Proxy settings
HTTP_PROXY=
HTTPS_PROXY=
NO_PROXY=localhost,127.0.0.1
```

Claude Desktop Integration

Configuration File Location

- **Windows**: `%APPDATA%\Claude\claude_desktop_config.json`
- **macOS**: `~/Library/Application Support/Claude/claude_desktop_config.json`
- **Linux**: `~/.config/Claude/claude_desktop_config.json`

Complete Configuration

```
{
  "mcpServers": {
```

```
    "database-toolbox": {
      "command": "D:\\repos\\mcp-toolbox\\toolbox.exe",
      "args": [
        "--tools-file",
        "D:\\repos\\mcp-toolbox\\toolsdb.yaml",
        "--stdio"
      ],
      "env": {
        "GOOGLE_APPLICATION_CREDENTIALS": "D:\\repos\\mcp-toolbox\\complete-tube-
421007-208a4862c992.json",
        "GOOGLE_CLOUD_PROJECT": "complete-tube-421007"
      }
    },
    "release-notes-server": {
      "command": "D:\\repos\\mcp-toolbox\\toolbox.exe",
      "args": [
        "--tools-file",
        "D:\\repos\\mcp-toolbox\\tools.yaml",
        "--stdio"
      ],
      "env": {
        "GOOGLE_APPLICATION_CREDENTIALS": "D:\\repos\\mcp-toolbox\\complete-tube-
421007-208a4862c992.json"
      }
    }
  }
}
```

Verifying Integration

1. **Restart Claude Desktop** completely (quit and reopen)
2. Look for the **hammer icon (🔨)** in the chat interface
3. Click the hammer to see available tools
4. Test with: "Search for hotels in New York"

Google ADK Agent Development

Installing ADK

```
# Install Google ADK
pip install google-adk google-genai-toolbox

# Install toolbox client
pip install toolbox-core
```

Hotel Booking Agent

gcp-hotel-agent/agent.py:

```
from google.adk.agents.llm_agent import Agent
from toolbox_core import ToolboxSyncClient
import os

# Initialize toolbox client
toolbox = ToolboxSyncClient("http://127.0.0.1:5000")

# Load tools from configuration
tools = toolbox.load_toolset('hotel-management')

# Configure the agent
root_agent = Agent(
    name="hotel_booking_assistant",
    model="gemini-2.0-flash",
    description="AI assistant for hotel search and booking",
    instruction="""
    You are a helpful hotel booking assistant with access to a database of hotels.
    You can:
    1. Search for hotels by name or location
    2. Show available hotels
    3. Make bookings with check-in/check-out dates
    4. Cancel existing bookings
    5. Provide booking details and calculate costs

    Always confirm booking details before making a reservation.
    Calculate total costs when showing booking information.
    Be helpful and suggest alternatives if requested hotels are unavailable.
    """,
    tools=tools,
    temperature=0.7,
    max_tokens=2048
)
```

Release Notes Agent

gcp-releasenotes-agent-app/agent.py:

```
from google.adk.agents.llm_agent import Agent
from toolbox_core import ToolboxSyncClient
import logging

# Setup logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

# Initialize connection
toolbox = ToolboxSyncClient("http://127.0.0.1:5000")

# Load release notes tools
tools = toolbox.load_toolset('release_notes_tools')
```

```
root_agent = Agent(
    name="gcp_release_notes_analyst",
    model="gemini-2.0-flash",
    description="Expert on Google Cloud Platform release notes and updates",
    instruction="""
    You are an expert analyst for Google Cloud Platform release notes.
    Your role is to:
    1. Provide information about recent GCP product updates
    2. Search for specific product release notes
    3. Summarize important changes and new features
    4. Help users understand the impact of updates

    When presenting release notes:
    - Group by product when showing multiple updates
    - Highlight breaking changes or deprecations
    - Mention the release date for context
    - Explain technical terms in simple language when needed
    """,
    tools=tools,
    response_format="markdown"
)
```

Running the Toolbox

Standalone Mode

```
# Run with specific tools file
./toolbox --tools-file="toolsdb.yaml"

# Run as HTTP server
./toolbox --tools-file="toolsdb.yaml" --port=5000

# Run with verbose logging
./toolbox --tools-file="toolsdb.yaml" --log-level=debug

# Run with multiple tool files
./toolbox --tools-file="toolsdb.yaml,tools.yaml"
```

STDIO Mode (for Claude Desktop)

```
# Required for MCP protocol communication
./toolbox --tools-file="toolsdb.yaml" --stdio
```

Docker Deployment

```
# Dockerfile
FROM golang:1.21-alpine AS builder

WORKDIR /app
COPY . .
RUN go build -o toolbox .

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/

COPY --from=builder /app/toolbox .
COPY *.yaml .
COPY service-account-key.json .

ENV GOOGLE_APPLICATION_CREDENTIALS=/root/service-account-key.json
EXPOSE 5000

CMD ["./toolbox", "--tools-file=toolsdb.yaml", "--port=5000"]
```

Build and run:

```
docker build -t mcp-toolbox .
docker run -p 5000:5000 \
  -v $(pwd)/service-account-key.json:/root/service-account-key.json \
  mcp-toolbox
```

Testing and Validation

Manual Testing

```
# Test tool configuration
./toolbox --tools-file="toolsdb.yaml" --validate

# Test specific tool
./toolbox --tools-file="toolsdb.yaml" --test-tool="search-hotels-by-name" --
param="name=Plaza"

# Interactive testing
./toolbox --tools-file="toolsdb.yaml" --interactive
```

Using MCP Inspector

```
# Install MCP Inspector
npm install -g @modelcontextprotocol/inspector
```

```
# Run inspector
mcp-inspector ./toolbox --tools-file="toolsdb.yaml" --stdio
```

Python Test Script

```
import requests
import json

# Test toolbox server
def test_hotel_search():
    url = "http://localhost:5000/tools/search-hotels-by-location"
    payload = {
        "parameters": {
            "location": "New York"
        }
    }

    response = requests.post(url, json=payload)
    assert response.status_code == 200

    data = response.json()
    print(f"Found {len(data['results'])} hotels in New York")
    return data

# Run test
if __name__ == "__main__":
    results = test_hotel_search()
    for hotel in results['results']:
        print(f"- {hotel['name']}: ${hotel['price_per_night']}/night")
```

Troubleshooting

Common Issues and Solutions

1. Claude Desktop Not Detecting MCP Server

Symptoms: No hammer icon appears in Claude interface

Solutions:

```
# Verify JSON syntax
python -m json.tool < claude_desktop_config.json

# Check absolute paths
realpath toolbox.exe
realpath toolsdb.yaml

# Test toolbox manually
./toolbox --tools-file="toolsdb.yaml" --stdio
```

```
# Check Claude logs (Windows)
type %APPDATA%\Claude\logs\mcp.log
```

2. Authentication Failures

Error: "Could not find default credentials"

Solutions:

```
# Verify service account key
gcloud auth activate-service-account --key-file=service-account-key.json

# Test authentication
gcloud auth application-default print-access-token

# Set credentials explicitly
export GOOGLE_APPLICATION_CREDENTIALS="$(pwd)/service-account-key.json"
gcloud config set project complete-tube-421007
```

3. BigQuery Permission Errors

Error: "Permission denied on table"

Solutions:

```
-- Grant necessary permissions
GRANT `roles/bigquery.dataViewer` ON SCHEMA `test`
TO "serviceAccount:mcp-toolbox-sa@complete-tube-421007.iam.gserviceaccount.com";

GRANT `roles/bigquery.dataEditor` ON TABLE `test.hotels`
TO "serviceAccount:mcp-toolbox-sa@complete-tube-421007.iam.gserviceaccount.com";
```

4. Connection Timeouts

Error: "Context deadline exceeded"

Solutions:

```
# Increase timeout in tools configuration
sources:
  my-bigquery-source:
    kind: bigquery
    project: complete-tube-421007
    timeout: 30s  # Increase timeout
```

```
    max_retries: 3
    retry_delay: 2s
```

5. Parameter Serialization Issues

Known Bug: Only first parameter-based MCP call succeeds (Issue #4192)

Workaround:

```
# Use named parameters consistently
parameters:
  - name: hotel_id
    type: integer
    description: Hotel ID
    required: true  # Mark required parameters
```

Debug Mode

Enable detailed logging:

```
# Set environment variables
export TOOLBOX_LOG_LEVEL=debug
export GOOGLE_CLOUD_ENABLE_LOGGING=true

# Run with debug output
./toolbox --tools-file="toolsdb.yaml" --log-level=debug --verbose 2>&1 | tee
debug.log
```

Security Best Practices

1. Service Account Management

```
# Use least privilege principle
gcloud iam roles create mcp_toolbox_role \
    --project=complete-tube-421007 \
    --title="MCP Toolbox Custom Role" \
    --permissions=bigquery.tables.getData,bigquery.tables.update

# Rotate keys regularly
gcloud iam service-accounts keys create new-key.json \
    --iam-account=mcp-toolbox-sa@complete-tube-421007.iam.gserviceaccount.com

# Delete old keys
gcloud iam service-accounts keys delete KEY_ID \
    --iam-account=mcp-toolbox-sa@complete-tube-421007.iam.gserviceaccount.com
```

2. Secure Configuration Storage

```
# Encrypt sensitive files
openssl enc -aes-256-cbc -salt -in service-account-key.json -out service-account-
key.enc

# Use secrets management
gcloud secrets create mcp-service-account \
    --data-file=service-account-key.json

# Access in application
gcloud secrets versions access latest --secret=mcp-service-account
```

3. Network Security

```
# Configure VPC Service Controls
sources:
  secure-bigquery:
    kind: bigquery
    project: complete-tube-421007
    vpc_service_control:
      perimeter: projects/12345/accessPolicies/policy/servicePerimeters/perimeter
    private_ip: true
```

4. Audit Logging

```
-- Enable BigQuery audit logs
CREATE OR REPLACE TABLE `audit.mcp_toolbox_logs` AS
SELECT
  timestamp,
  protoPayload.authenticationInfo.principalEmail as user_email,
  protoPayload.methodName as operation,
  protoPayload.resourceName as resource,
  protoPayload.request as request_details
FROM
  `complete-tube-421007.cloud_audit_logs.data_access`
WHERE
  protoPayload.serviceName = 'bigquery.googleapis.com'
  AND protoPayload.authenticationInfo.principalEmail LIKE '%mcp-toolbox%';
```

Examples

Example 1: Hotel Search and Booking Flow

```
# Complete booking workflow
async def book_hotel_workflow():
    # Search for hotels
    hotels = await toolbox.execute_tool(
        "search-hotels-by-location",
        {"location": "Miami Beach"}
    )

    # Select a hotel
    selected_hotel = hotels[0]

    # Make booking
    booking_result = await toolbox.execute_tool(
        "book-hotel",
        {
            "hotel_id": selected_hotel["id"],
            "checkin_date": "2024-12-25",
            "checkout_date": "2024-12-30"
        }
    )

    # Get confirmation
    details = await toolbox.execute_tool(
        "get-booking-details",
        {"hotel_id": selected_hotel["id"]}
    )

    return details
```

Example 2: Multi-Database Query

```
# Advanced multi-source configuration
sources:
  bigquery-prod:
    kind: bigquery
    project: production-project

  cloudsql-analytics:
    kind: postgres
    host: ${CLOUDSQL_HOST}
    database: analytics

  firestore-cache:
    kind: firestore
    project: complete-tube-421007
    database: cache

tools:
  combined-search:
    kind: custom
```

```
    sources:
      - bigquery-prod
      - cloudsql-analytics
      - firestore-cache
    description: Search across all data sources
    implementation: |
      # Custom logic to query multiple sources
      # and combine results
```

Contributing

We welcome contributions! Please follow these guidelines:

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/amazing-feature`)
3. Commit your changes (`git commit -m 'Add amazing feature'`)
4. Push to the branch (`git push origin feature/amazing-feature`)
5. Open a Pull Request

Development Setup

```
# Clone repository
git clone https://github.com/yourusername/mcp-toolbox.git
cd mcp-toolbox

# Install development dependencies
pip install -r requirements-dev.txt

# Run tests
pytest tests/

# Run linting
black .
flake8 .
```

License

This project is licensed under the Apache License 2.0 - see the LICENSE file for details.

Acknowledgments

- Google Cloud Platform for BigQuery and cloud services
- Anthropic for the Model Context Protocol specification
- Google AI Development Kit for agent framework
- MCP community for toolbox development and support

Support

For issues and questions:

- GitHub Issues: Report bugs or request features
- Discord: MCP Community
- Documentation: Official MCP Docs
- Google Cloud Support: BigQuery Documentation
