AgriConnect

- Project Sheet
- How to diagnose issues if something fails

Project Sheet

Name	AgriConnect
Product Description	An AI-powered chatbot that helps extension officers support farmers at scale
Repository Link	https://github.com/akvo/rag-doll
Product Documentation	
Tech Stack	List of technologies used to execute the technical scope of the project: • Front • Back • BD • Testing • CI • Hosting
Asana Link	https://app.asana.com/0/1207876388691339/1209041847057525
Slack Channel Link	proj-agriconnect-general

How to diagnose issues if something fails

1. Health Check Endpoint

Before diagnosing issues, it is important to verify the health status of the containers. The Agriconnect platform provides an endpoint to check the health of its services:

> Health Check URL: https://agriconnect.akvotest.org/api/health-check

Purpose

The health check endpoint returns the status of the containers, indicating whether they are up and running. A successful health check implies that the services are operational.

Expected Response

When all containers are healthy, the endpoint returns a JSON response similar to the following:

```
{
  "status": "healthy",
  "services": {
    "rabbitmq": true,
    "chromadb": true,
    "database": true,
    "assistant": true,
    "eppo-librarian": true,
    "backend": true
  }
}
```

Response Details

- status: Indicates the overall health of the containers. If all services are operational, the status will be "healthy".
- services: A list of services with their respective health status. A value of true means the service is running, while false indicates a problem with that service.

2. Diagnosing Issues

If the health check indicates that one or more services are not operational (i.e., the status of a service is false), follow these steps to diagnose and resolve the issue:

a. Identify the Affected Service(s)

Review the JSON response from the health check to pinpoint which services have a false status.

b. Check Logs

Access the logs for the affected service(s) to identify any error messages or anomalies that may indicate the cause of the issue. Since we use Google Cloud, you will need to check the logs using the Google Cloud Console. Access Google Cloud Console:

- Go to Google Cloud Console.
- Navigate to Kubernetes Engine > Workloads.
- Filter Logs:
  - Use the Cluster and Namespace filters to locate the relevant logs.
  - Filter by the test cluster and agriconnect-namespace namespace to view the specific workloads.

AD_4nXfmt-1MxIk4GD6pLT6E-vrlIdeZ9eCC8BJicuzqda0x36j0Zi5ZAY9W_vYaisSXLoUF6d_bq69S1pkPt

Within the workloads, identify the container with an unhealthy status. Select this container and then navigate to the "Logs" tab to view its logs. Analyzing these logs can provide insights into the issue, such as error messages or patterns that suggest the cause. A common issue might be related to an unstable Socket.IO connection, but the logs will help pinpoint the specific problem. Use this information to begin debugging and addressing the issue.

c. Restart the Service

If the issue is not apparent from the logs, restarting the affected service(s) may resolve the problem. This can be done by deleting the pods associated with the unhealthy container.

AD_4nXflTUKiyVklVQ7fxpI1SVh-gaq8i35BRzhltyMsaZg_AKV24Iq4id6D2BSOqYszMAASTIluUe3iajDTqJ

In the Google Cloud Console, navigate to the "Manage Pods" section for the specific container that is reported as unhealthy. Within this section, you can access the "Pods Detail" view. By deleting the pod from this view, the container will automatically restart, which may help restore it to a healthy state. This process is often effective in clearing transient issues that cause service disruptions.

AD_4nXd0UJBkmtuJqhHKoQOSb3AxQtz5ZIyo_i2_WN-tNG_lfy1lhNO2asjSxJTPAkMh_AEZsDUeapNViIau

3. Most Common Issues

Below are examples of some common issues that may occur on the Agriconnect platform, along with their related containers and explanations.

Issue	Related Containers	Explanation
Assistant Messages/Whispers Not Delivered to Extension Officer Chat Window	• agriconnect-rabbitmq • assistant-deployment • backend-deployment	• This issue often arises when messages fail to be delivered between the assistant and backend containers via RabbitMQ. It can also occur if the backend fails to send the assistant's message to the Extension Officer via Socket.IO. • If all three containers (RabbitMQ, assistant, and backend) are up and running, RabbitMQ will automatically resend the message. • The backend has a recheck status feature that will attempt to resend undelivered messages to the Extension Officer. • If any of these containers are down, a restart may be necessary (refer to section c under point 2 for instructions on restarting containers).

Issue	Related Containers	Explanation
Extension Officer Messages Not Delivered to Farmer Device	• backend-deployment	• This issue can occur due to a Socket.IO connection problem between the frontend (FE) and backend (BE), or it could be related to a failure in the Twilio service to send the message. • If a message cannot be sent via Socket.IO from the FE to the BE, the frontend has a resend method that will attempt to resend failed messages (saved in IndexedDB) to the backend when the Socket.IO connection is reestablished. • It is also important to review the Twilio message logs to determine the cause of the failure. You can access Twilio logs at Twilio Console.
Farmer Messages Not Delivered to Extension Officer Chat Window	• backend-deployment	• This issue can occur for similar reasons as the "Extension Officer Messages Not Delivered to Farmer Device" problem. It is often related to issues with the Twilio webhook or the Socket.IO connection. • The backend includes a recheck status feature designed to automatically attempt to resend any undelivered messages to the Extension Officer. This feature helps ensure that messages are eventually delivered, even if initial delivery attempts fail due to connectivity issues.