Software Release Note – Chalkbot v1.0.1

Release note reference: P5056-SRN-001 v1.0.1

# Overview

|  |  |
| --- | --- |
| Product | Chalkbot Proof of Concept – phone to GPT service, allowing users to access Open AI’s GPT Large Language Models via a phone number and without the need for data connection to the internet |
| Version | v1.0.1 |
| Date | 7th December 2023 |
| Location | Code repository stored in Cambridge Consultants Internal Gitlab, shared as a zip file chalkbot-v1.0.1.zip |
| Type | Proof-of-concept |

# Release Contents

|  |  |
| --- | --- |
| Directory | Description |
| Deploy | Contains instructions for deploying the service using a local machine (as opposed to the intended gitlab pipelines). |
| Infrastructure | Contains terraform code used for deployment of main AWS services. |
| Python | Contains the python code used for the lambda functions. |
| Gitlab-ci.yml | Contains the code responsible for the jobs in the gitlab pipeline, used for linting, building and deployment. |
| README.md | Describes key information about the code such as deployment instructions using gitlab. Instructions for development and an architecture diagram. |
| LICENSE.md | Licensing information |

# Referenced Documents

|  |  |  |  |
| --- | --- | --- | --- |
| Document | Type | Location | Date |
| P5056-C-001 v1.0 | Memorandum of Agreement | Docman P5056-C-001 v1.0 and included in chalkbot-v1.0.0.zip | 09/11/2023 |
| P5056-M-005 v1.0 | Schedule of Technology with a Restrictive Encumbrance | Docman P5056-M-005 v1.0 and included in chalkbot-v1.0.0.zip | 13/12/2023 |

# Licensing Information

CAMBRIDGE LICENSES THE SOFTWARE "AS IS," AND MAKES NO EXPRESS OR IMPLIED WARRANTY OF ANY KIND. CAMBRIDGE SPECIFICALLY DISCLAIMS ALL INDIRECT OR IMPLIED WARRANTIES TO THE FULL EXTENT ALLOWED BY APPLICABLE LAW, INCLUDING WITHOUT LIMITATION ALL IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, TITLE OR FITNESS FOR ANY PARTICULAR PURPOSE.

THE SOFTWARE IS DELIVERED IN ACCORDANCE WITH TERMS AND CONDITIONS SET OUT IN THE AGREED MEMORANDUM OF AGREEMENT (P5056-C-001 V1.0) AND THE SCHEDULE OF TECHNOLOGY WITH A RESTRICTIVE ENCUMBRANCE (P5056-M-005 V1.0).

# Installation Instructions

The POC was developed and deployed using GitLab pipelines however a method for deploying on a local machine is also included. These can be found in the high level README and the README found in the deploy directory respectively but are also included below. 5.2 backend setup must first be followed whichever method you use.

## AWS Region

The AWS region used to deploy chalkbot is currently set as eu-west-2 (London). This can easily be changed by changing both the terraform init backend-config command and the [AWS configuration](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html) to allow you to deploy the service to any desired AWS region.

Since the chalkbot POC was for London, the phone number used is currently set to a UK number, this can also easily be updated in infrastructure/connect/main.tf’s aws\_connect\_phone\_number resource to a number from any country allowed by Amazon Connect.

## Backend Setup (Both for Gitlab and Local Machine Deployment)

It is assumed for this step you have AWS CLI installed and configured correctly.

Certain AWS components used must be globally unique. Since the values in the source code are already taken by the chalkbot POC deployment, you need to:

Update the S3 bucket name used to store Terraform State to be unique:

1. Update the TERRAFORM\_STATE\_BUCKET\_NAME variable to be a globally unique name of your choice in deploy/scripts/backend\_setup.sh line 15.
2. Update the backend-config bucket in deploy/scripts/terraform\_aws\_cli.sh line 15 to match this.

Update the S3 bucket named used to store project resources to be unique:

1. Update the S3 bucket name in infrastructure/s3/main.tf line 9 to be a globally unique name of your choice.

Update the ECR repository name:

1. Update the ECR\_REPOSITORY\_NAME variable in deploy/scripts/backend\_setup.sh line 27 to be a globally unique name of your choice.
2. Update the ecr\_repository\_name in infrastructure/lambda/gpt-lambda/main.tf locals line 13 to match this.

Update the connect instance to be globally unique:

1. Update the aws\_connect\_instance instance\_alias in infrastructure/connect/main.tf to be a globally unique name of your choice.

Once this is done, running `./deploy/scripts/backend\_setup.sh` will set up the necessary ECR repository and Terraform backend.

Finally, since the AWS lambda function that makes the OpenAI API requests is deployed using an ECR image as its zip file is too large:

1. In the Push GPT Lambda Container to ECR GitLab CI job (or deploy/scripts/push\_gpt\_lambda\_to\_ecr.sh for local machine deployment), update the following variables accordingly: ECR\_URL, REPOSITORY\_NAME, IMAGE\_TAG where the ECR\_URL matches the URL and repository name created in the backend setup.
2. In infrastructure/lambda/gpt-lambda/main.tf, update the local variables ecr\_repository\_name and gpt\_lambda\_image\_tag to match these.

## Gitlab Deployment

There are several jobs in the pipeline responsible for the deployment of the service. The pipeline assumes that the ECR repository and terraform backend have already been created (as above) as these are not managed by the Terraform infrastructure code. Once this is done:

The following Gitlab CI/CD environment variables need to be set:

* AWS\_CONFIG - config file as described in [AWS CLI documentation](https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-files.html)
* AWS\_CREDENTIALS - credentials file as described in [AWS CLI documentation](https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-files.html)
* CONNECT\_ADMIN\_PASSWORD - the password that you want associated with your Amazon Connect instance admin user account
* OPENAI\_API\_KEY - key required to use OpenAI's API

Running the pipeline on the main branch should then result in the service being deployed.

The service should now be deployed. To view the phone number, log in to your connect instance, go to channels/phone numbers and the number will be there.

## Local Machine Deployment

### Requirements

The following are required for the installation of chalkbot:

* bash
* Docker
* [Terraform](https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli) v1.5.6
* [aws-cli](https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html)
* jq
* zip

### Steps

Once the backend has been set up (as above), the following need to be done:

1. Update OPENAI\_API\_KEY in deploy/scripts/terraform\_aws\_cli.sh line 20 to be your OpenAI API key.
2. Update CONNECT\_ADMIN\_PASSWORD in deploy/scripts/terraform\_aws\_cli.sh line 21 to be your desired password for the Amazon Connect instance admin user account.
3. Run ./deploy/deploy.sh from within the chalkbot repository directory (i.e. not from within the deploy directory)

The service should now be deployed. To view the phone number, log in to your connect instance, go to channels/phone numbers and the number will be there.

# Known Issues

## Response Time

It is known that it can sometimes take 10+ seconds to hear a response after asking a question. This is due to a combination of the time it takes for Amazon Lex to convert your question from speech to text and the time it takes for OpenAI’s API to give a response – which can take particularly long for more complex questions. Unlike with the GUI on chat.openai.com, the API does not stream the response token-by-token (back to the Lambda where the API request is made) but waits until the entire response has been generated and returns it all in one go.

Using a faster model than gpt-3.5-turbo may help with this. There may also be some workaround to enable the response to be streamed to the lambda and have this streamed back to Amazon Lex, however due to the nature of AWS Lambda functions this would likely be quite complex to implement and is not required for the POC. We created a lambda to periodically trigger Amazon Lex to help address the cold start issue of slow speech-to-text conversion for a caller’s first question.

## Missing Terraform Components for Amazon Lex v2 and Amazon Connect

There are two important actions that are not currently available through Terraform, associating an Amazon Lex v2 bot with an Amazon Connect instance, and associating an Amazon Connect contact flow with a phone number so these actions had to be done by passing the output from Terraform apply into aws-cli commands.