Google Cloud

Taking Your Chatbot to Production

Agenda

Operationalizing Your Agent

Deploying a WebHook for Fulfillment

Building a Custom Chatbot User Interface

Securing the Webhook

Integrations

Operationalizing your agent

Automation

Leverage existing sources like playbook, FAQs, etc. to build your agent.

Backend services

Connect to backend systems and services.

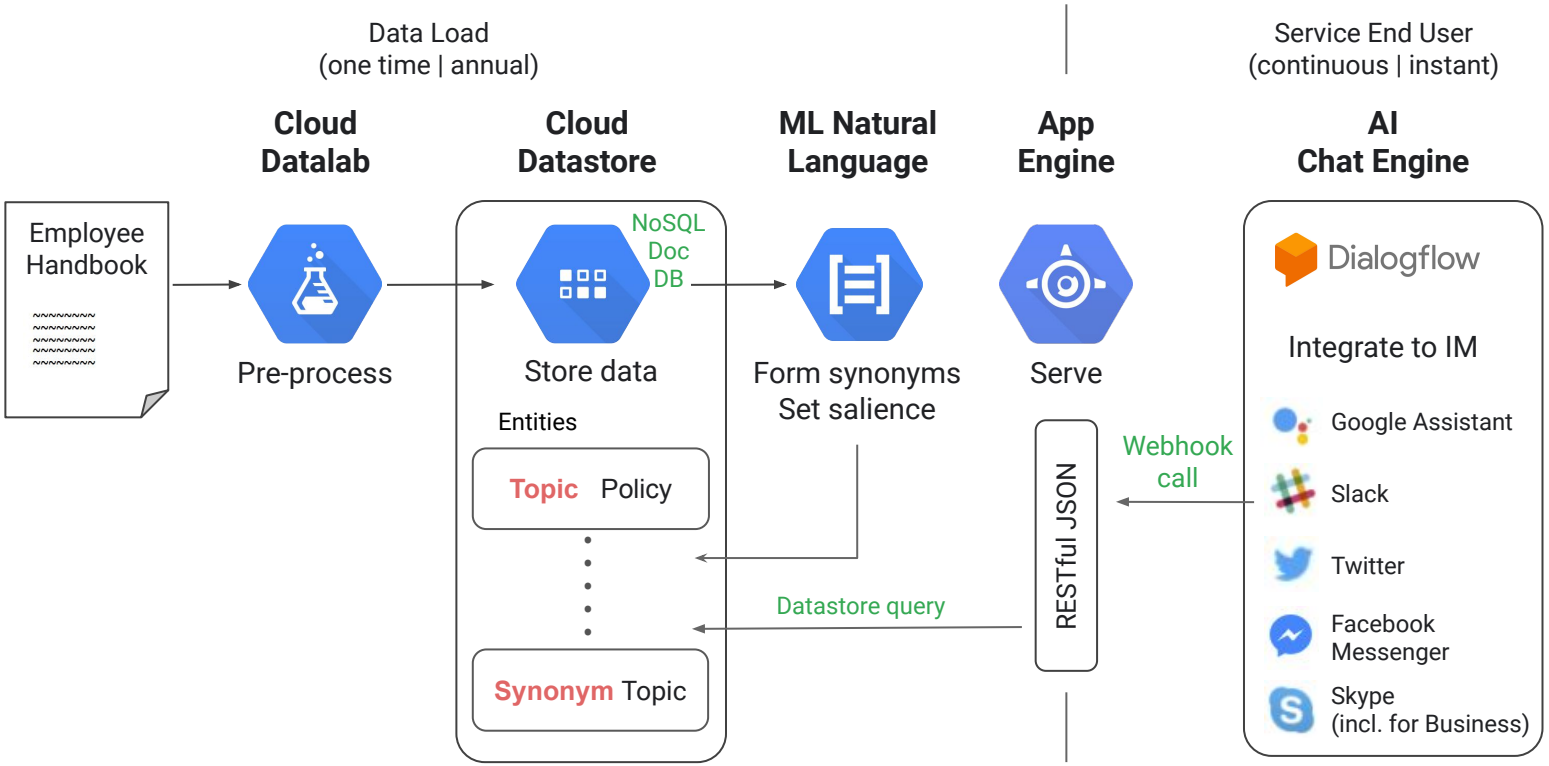
Branding

Customize your frontend to include branding, logos, etc.

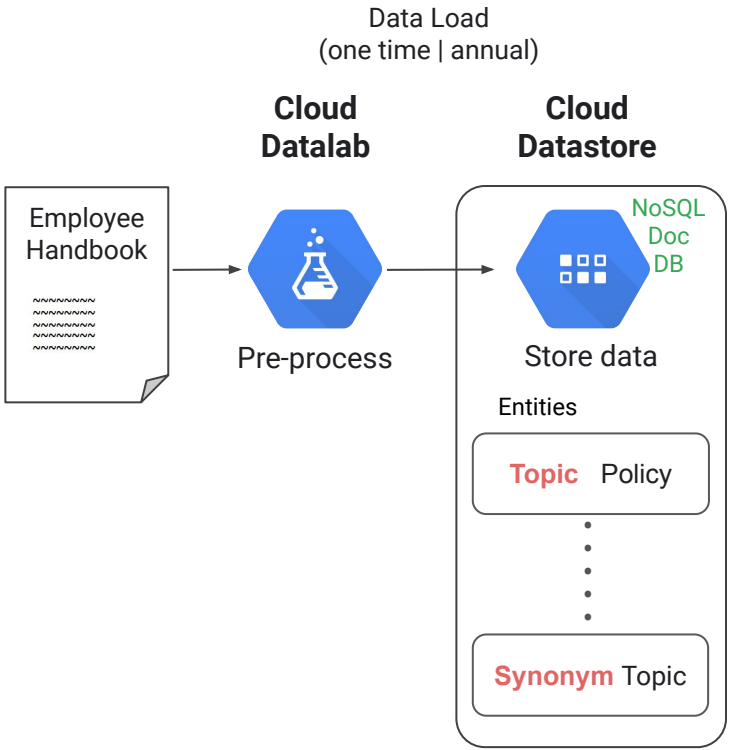
Security

Secure your webhook to allow authenticated calls only.

HR chatbot architecture



Leveraging existing sources to derive entities



Cloud Datastore is a good place to store and retrieve your chatbot's data with minimal fuss!

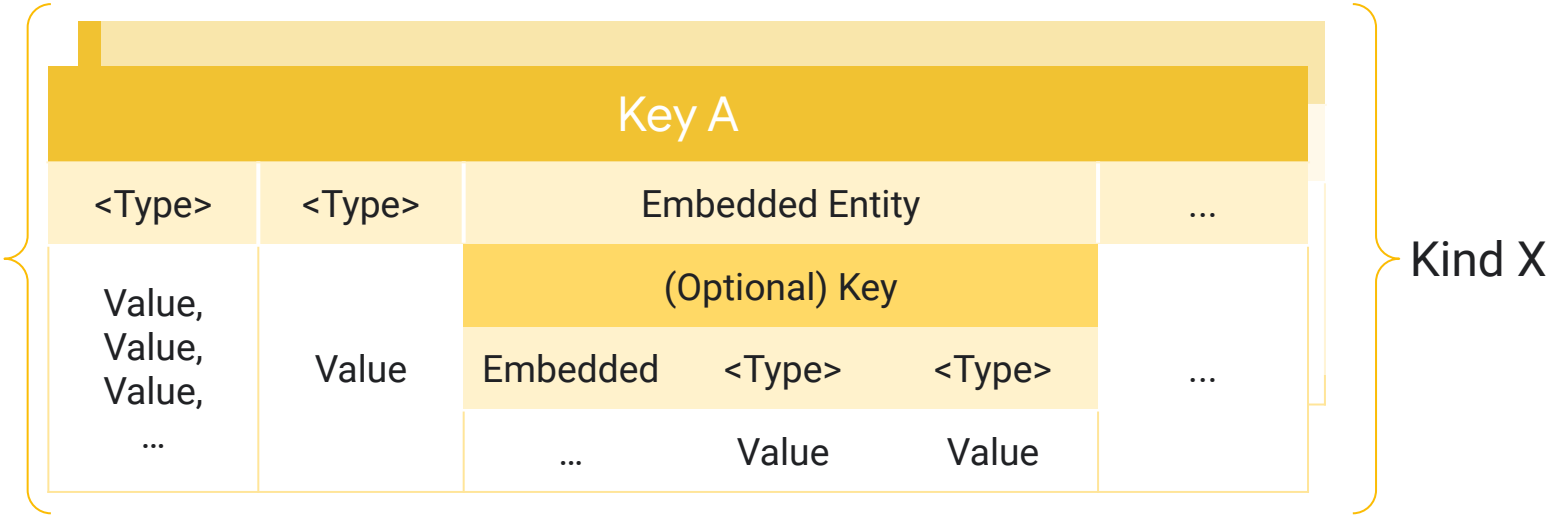
Cloud Datastore is a serverless global document database.

Database as a service

Features

Planet-scale
7 years+
15 trillion requests/month
Fully managed service

Cloud Datastore data model



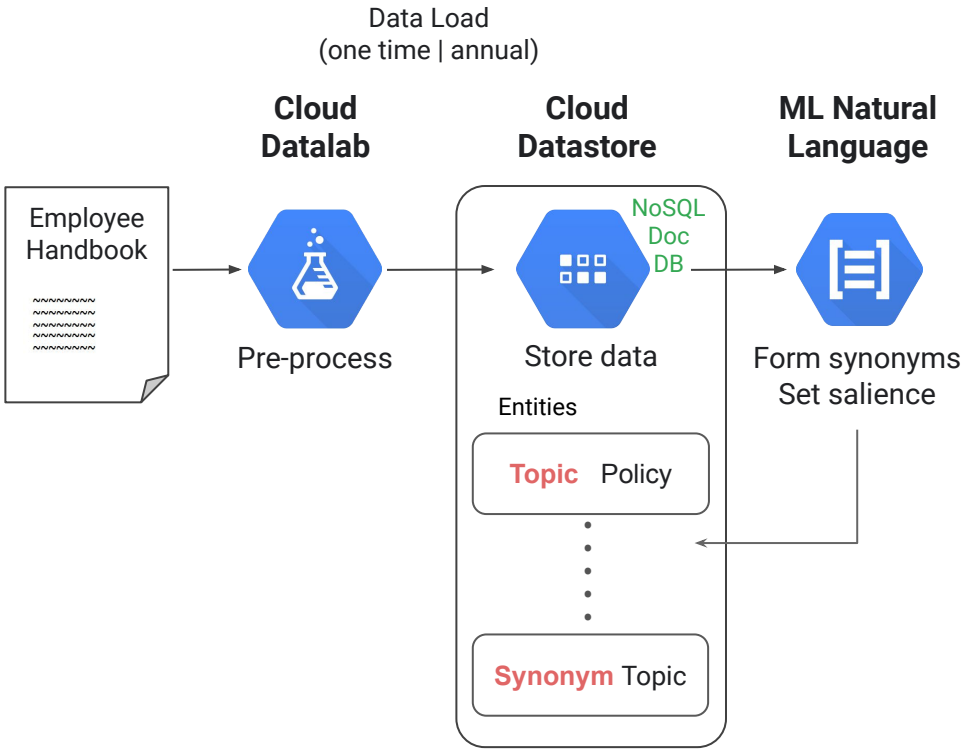
Deploying to Cloud Datastore

Google Cloud Datalab ProcessData.ipynb

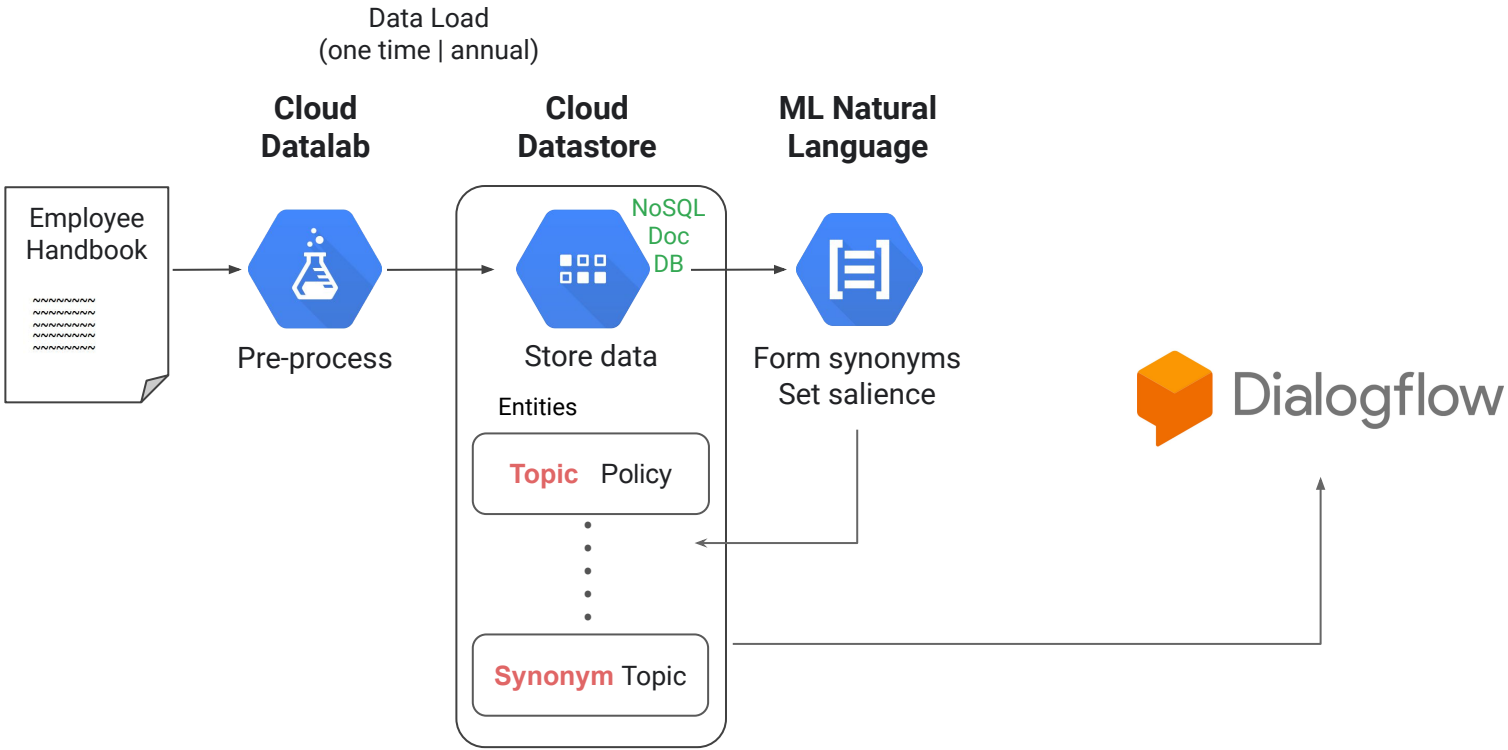
NOTEBOOK CODE TEXT DELETE MOVE UP MOV

```
from google.cloud import datastore
import json

datastore_client = datastore.Client(project='my-project')
kind = 'Word'

  for line in f:
    i = i + 1
    json_object = json.loads(line)
    word = json_object['word']
    word_key = datastore_client.key(kind, word)
    word_entity = datastore.Entity(key=word_key)
    word_entity['data'] = line
    words.append(word_entity)
  datastore_client.put_multi(words)
```

Adding ML to augment your agent



Populate entities into Dialogflow



Demo

Operationalizing Your Agent

In this demo, we'll use Cloud Datalab notebooks to quickly run Python scripts to extract topics from the sample HR Manual, then push them into a Datastore entity using the Cloud Datastore API. We also leverage the Natural Language API to generate synonyms for the HR topics. Finally, we use the Dialogflow API to populate entities into Dialogflow.

Agenda

Operationalizing Your Agent

Deploying a WebHook for Fulfillment

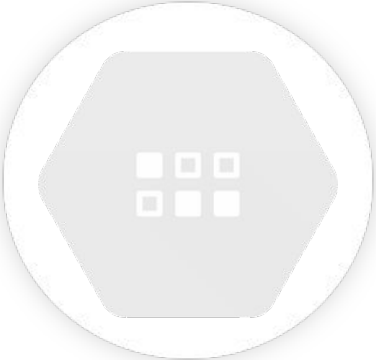
Building a Custom Chatbot User Interface

Securing the Webhook

Integrations

Understand the serverless options for your webhook

App Engine



Cloud Datastore

Cloud Functions

App Engine is a microservices platform for multiple programming languages

Event-oriented architectures/Webhook

Platform as a service

Triggers

HTTP

Language support

Code using popular languages, frameworks, and tools

Popular languages

Python
Java
PHP
Go
Node.js
C#
.NET

Popular frameworks

Django
Flask
Spring
webapp2
web2py

Popular tools

Eclipse
IntelliJ
PyCharm
Jenkins

Deploying to App Engine

In production

```
gcloud app deploy
```

In development

```
dev_appserver.py app.yaml (Python)
dev_appserver.sh (Java)
... o
```

app.yaml

```
application guest-book-123
version: 1
runtime: python27
api_version: 1

handlers:
- url: /favicon\.ico
  static_files: favicon.ico
  upload: favicon\.ico

- url: /.*
  script: main2.app

libraries:
- name: webapp2
  version: "2.5.2"
- name: jinja2
  version: "latest"
```

Understand the serverless options for your webhook

App Engine

Cloud Datastore

Cloud Functions

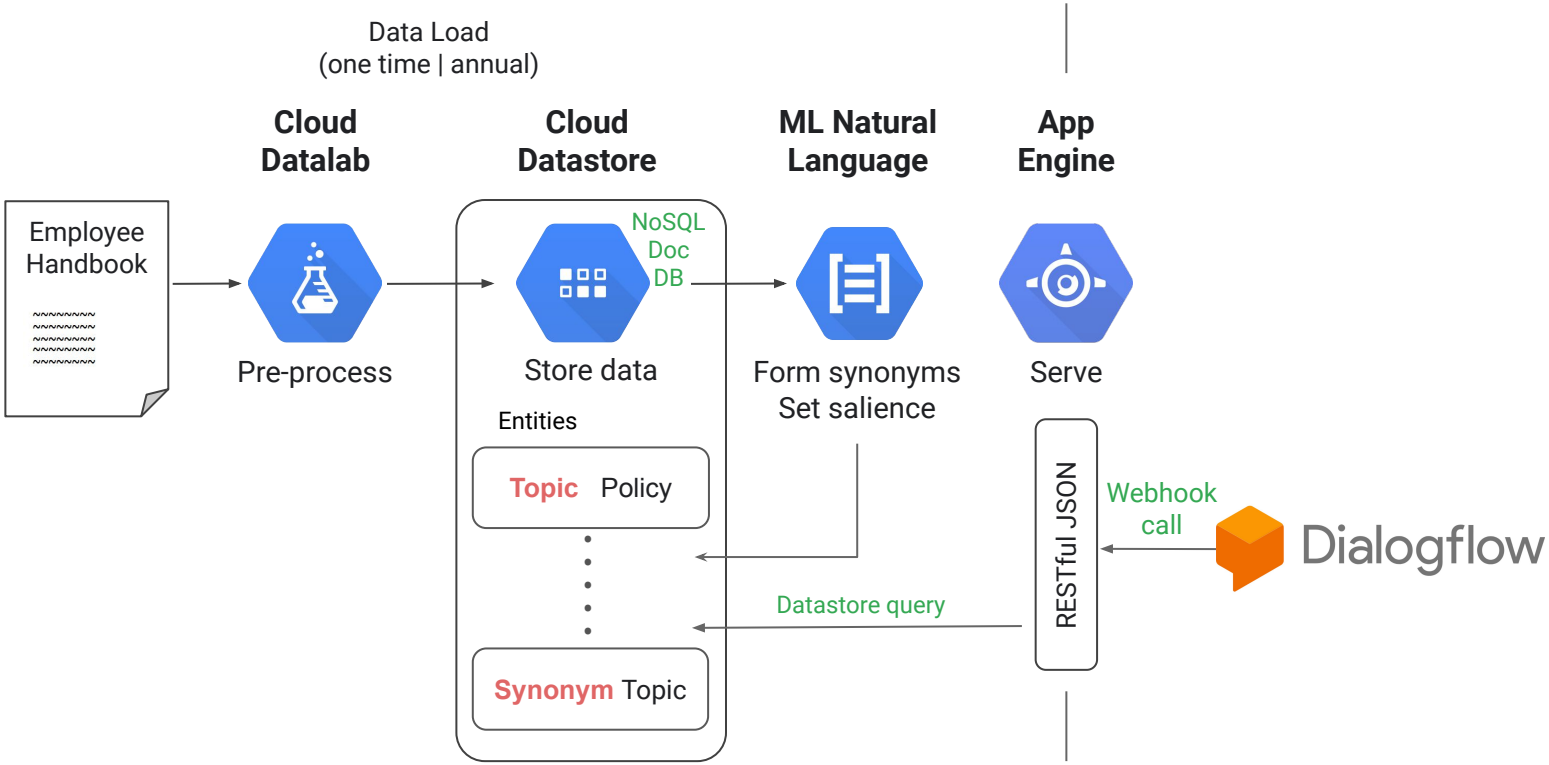
Understand the serverless options for your webhook

App Engine

Cloud Datastore

Cloud Functions

Deploying a webhook on App Engine



Use NDB to query Cloud Datastore synonyms

```
def getSynonym(query_text):
    synonym_key = ndb.Key('Synonym', query_text)
    synonyms = Synonym.query_synonym(synonym_key).fetch(1)

    synonym_text = ""
    for synonym in synonyms:
        synonym_text = synonym.synonym
        break

    return synonym_text
```

Use NDB to look up Topic Key in Cloud Datastore

```
def getActionText(synonym_text):
    synonym_text = synonym_text.strip()
    topic_key = ndb.Key('Topic', synonym_text)
    topics = Topic.query_topic(topic_key).fetch(1)

    action_text = ""
    for topic in topics:
        action_text = topic.action_text

    if action_text == None or action_text == "":
        return ""

    return action_text
```

Map data returned from Datastore into Python classes

```
class Topic(ndb.Model):
    action_text = ndb.StringProperty()

    @classmethod
    def query_topic(cls, ancestor_key):
        return cls.query(ancestor=ancestor_key)

class Synonym(ndb.Model):
    synonym = ndb.StringProperty()

    @classmethod
    def query_synonym(cls, ancestor_key):
        return cls.query(ancestor=ancestor_key)
```

app.yaml for backend service

```
runtime: python27
api_version: 1
threadsafe: true
service: dialogflow

# [START handlers]
handlers:
- url: /static
  static_dir: static
- url: /.*
  script: main.app
# [END handlers]
```

Agenda

Operationalizing Your Agent

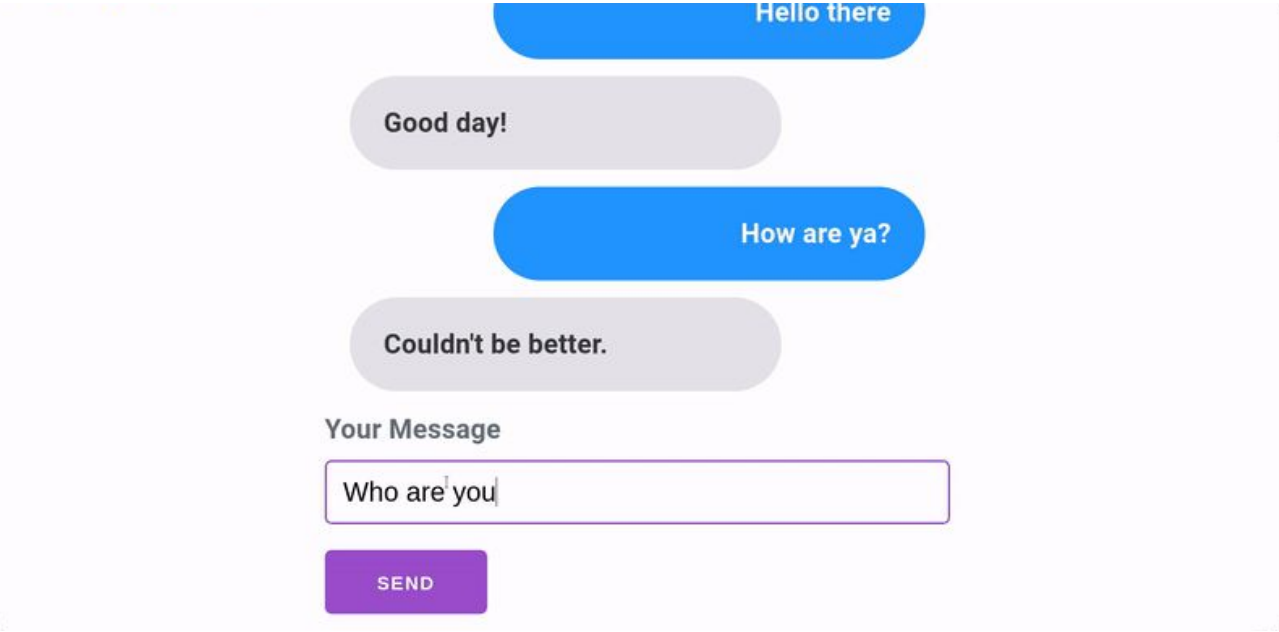
Deploying a WebHook for Fulfillment

Building a Custom Chatbot User Interface

Securing the Webhook

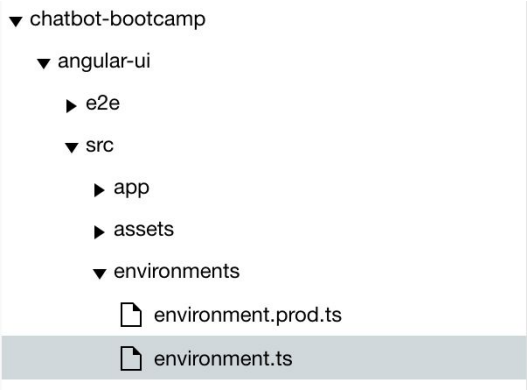
Integrations

Building a custom chatbot user interface



https://github.com/AngularFirebase/59-angular-chatbot-dialogflow

Building a custom chatbot user interface




```
export class ApiAiClient {
    constructor(options) {
        if (!options || !options.accessToken) {
            throw new ApiAiClientConfigurationError("Access token is required for new ApiAi.Client instance");
        }
        this.accessToken = options.accessToken;
        this.apiLang = options.lang || ApiAiConstants.DEFAULT_CLIENT_LANG;
        this.apiVersion = options.version || ApiAiConstants.DEFAULT_API_VERSION;
        this.apiBaseUrl = options.baseUrl || ApiAiConstants.DEFAULT_BASE_URL;
        this.sessionId = options.sessionId || this.guid();
    }
```

Dialogflow Javascript Client - ApiAiClient class

The ChatService class: chat.service.ts

```
@Injectable()
export class ChatService {



 readonly token = environment.dialogflow.angularBot;
 readonly client = new ApiAiClient({ accessToken: this.token });
```

app.yaml to deploy custom UI as default service

```yaml
runtime: python27
api_version: 1
threadsafe: true

skip_files:
- ^(?!dist)  # Skip any files not in the dist folder

handlers:
# Routing for bundles to serve directly
- url: /((?:inline|main|polyfills|styles|vendor)\.[a-z0-9]+\.bundle\.js)
 secure: always
 redirect_http_response_code: 301
 static_files: dist/\1
 upload: dist/.*
```

Demo

Deploying Webhook and Frontend on App Engine

In this demo, we'll deploy a custom UI and the webhook code on App Engine. At the end, we will be able to test our agent.

Lab

Lab 2,Part 1, 2

Agenda

Operationalizing Your Agent

Deploying a WebHook for Fulfillment

Building a Custom Chatbot User Interface

Securing the Webhook

Integrations

Require basic HTTP authentication to access

```
def requires_auth(f):
    @wraps(f)
    def decorated(*args, **kwards):
        auth = request.authorization
        if not auth or not check_auth(auth.username,
auth.password):
            return authenticate()
        return f(*args, **kwards)
    return decorated
```

Validate submitted username and password

```
def check_auth(username, password):
    """This function is called to check if a username / password
    combination is valid.
     """
     uname="myuser"
     pwd="mypassword"

     return username == uname and password == pwd
```

Handling authentication failure

```
def authenticate():
    """Sends a 401 response that enables basic auth"""
    logging.info("inside authenticate")
    return Response(
    'Could not verify your access level for that URL.\n'
    'You have to login with proper credentials', 401,
    {'WWW-Authenticate': 'Basic realm="Login Required"'})

@app.route('/webhook/', methods=['POST'])
@requires_auth
#def handle():
```

Demo

Adding Basic Authentication and Configuring Credentials

In this demo, we'll add basic authentication to our webhook and configure the credentials in the Dialogflow console.

Lab

Lab 2,Part 3

Agenda

Operationalizing Your Agent

Deploying a WebHook for Fulfillment

Building a Custom Chatbot User Interface

Securing the Webhook

Integrations

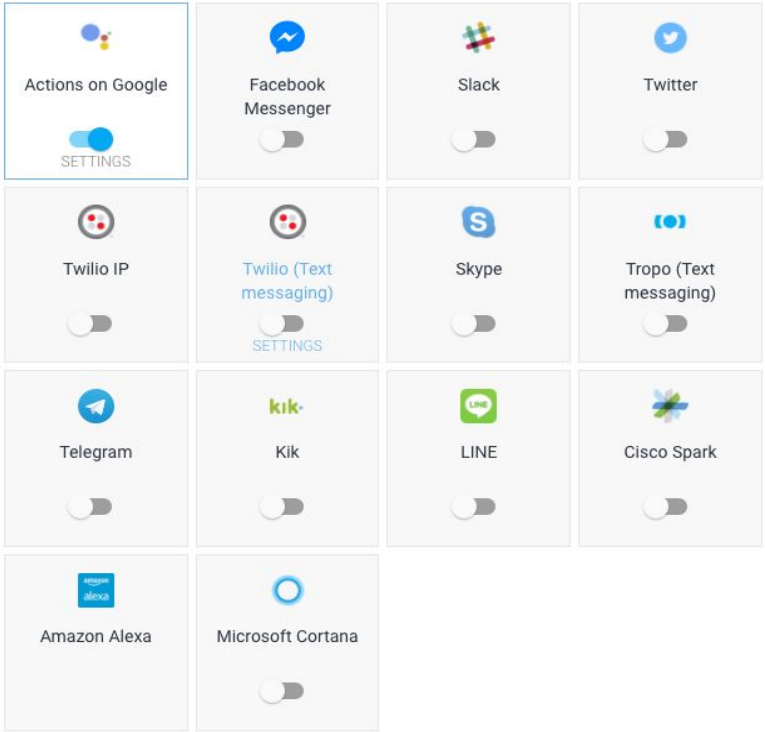
Dialogflow agents can be enabled in multiple channels and surfaces

Actions on Google

Google Home, Pixel, and more to come

External integrations

Slack, Facebook Messenger, Twitter, Twilio, Skype, Tropo, Telegram, Kik, LINE, Cisco Spark, Alexa, Cortana



Integrate seamlessly between surfaces with the Actions SDK

Google Home

Google Assistant

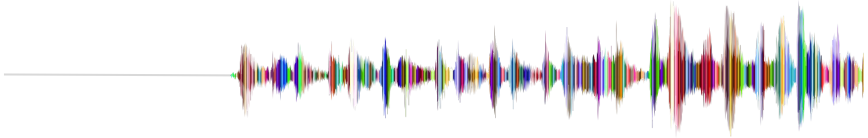
Dialogflow

Actions SDK

Demo

Building Voice Chat with Actions on Google

In this demo, we'll integrate the Dialogflow agent into an Actions on Google project to enable voice chat functionality on any Google Assistant device.



Demo

Integrating with Slack

In this demo, we'll integrate the Dialogflow agent within the Slack org to create a Dialogflow-powered Slackbot.

Summary

1. Set up a knowledge base on Cloud Datastore
2. Automate parts of your agent building process
3. Build a custom UI and set up a webhook on App Engine
4. Secure the webhook

cloud.google.com