AI with IBM Watson

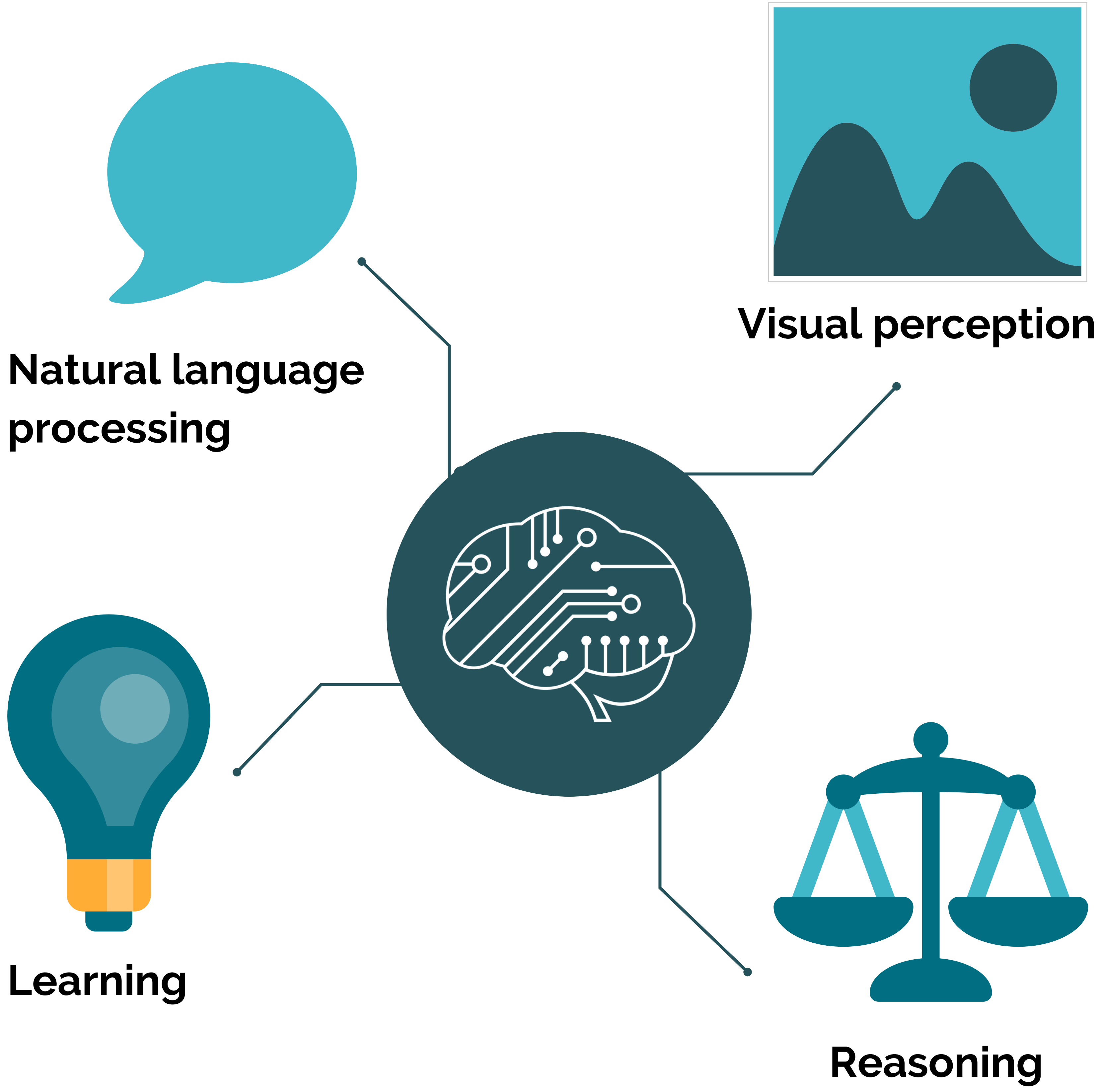
DEVINAR

presented by devleague

What is artificial intelligence (AI)?

- Branch of computer science focused on developing computer systems that simulate natural intelligence

- Further subdivided into machine learning (ML) and deep learning (DL)



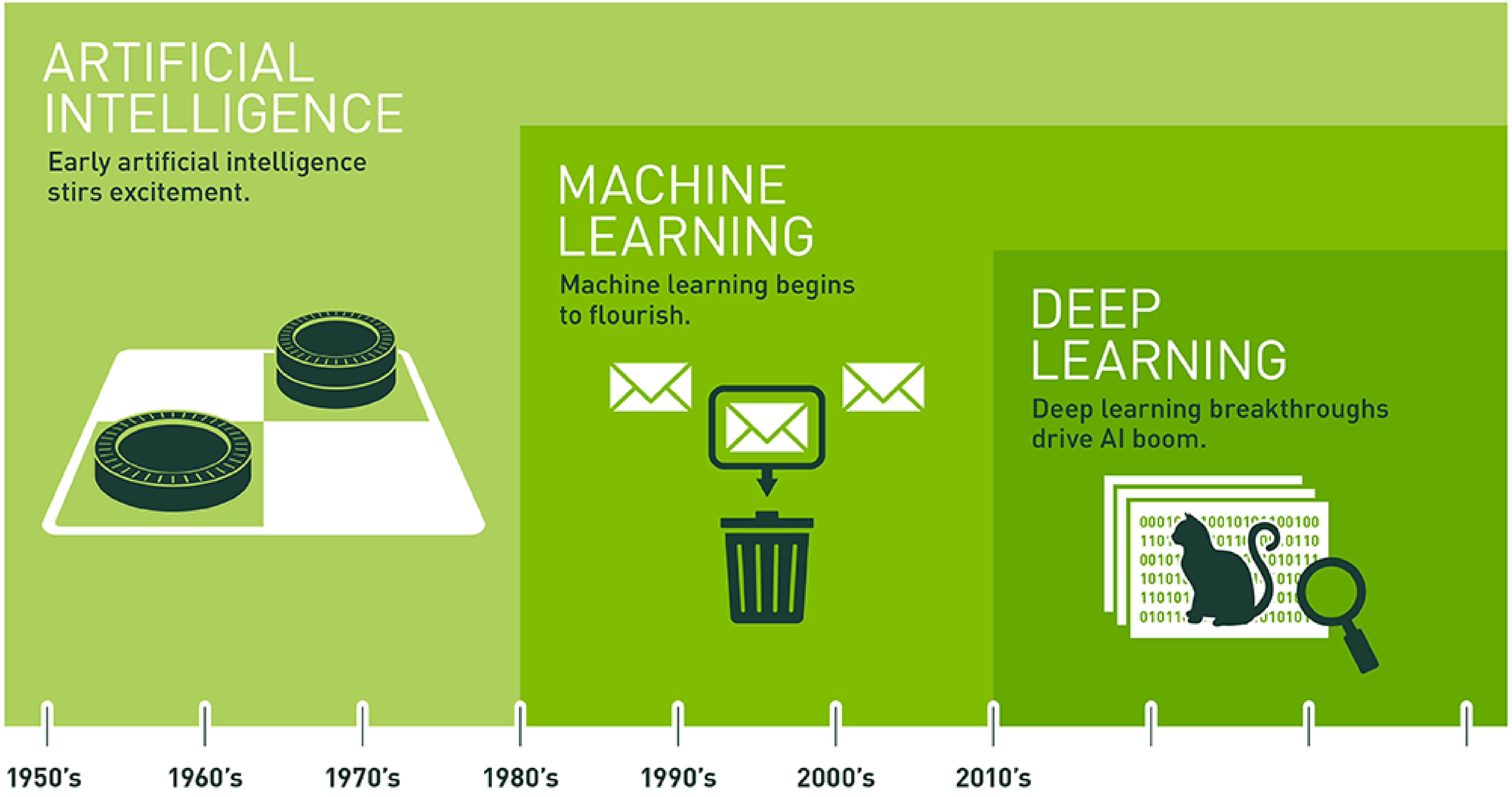
Natural language processing

Visual perception

Learning

Reasoning

Development of AI into subcategories


ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

The Difference Between AI, Machine Learning, and Deep Learning? | NVIDIA Blog. (2018, December 21). Retrieved from https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

Machine learning vs. deep learning

Machine Learning

- Consist of algorithms that parse data and learn from data to make informed decisions

e.g. **IBM Watson**, Netflix recommendations engine

Deep Learning

- Continuously analyzes data in a way that mimics the human brain
- Uses a layered structure of algorithms called an artificial neural network

e.g. Google's AlphaGo, Tesla Auto Pilot

Example

- Radio that increases volume whenever you say "Louder"

- Radio learns to be louder with other cues like "I can't hear"

AI with IBM Watson - 2011 Jeopardy!

- IBM Watson played Jeopardy! against top players and won

- AI able to understand natural language and make sense of vast amounts of information

Markoff, J. (2011, February 16). On 'Jeopardy!? Watson Win Is All but Trivial. Retrieved from https://www.nytimes.com/2011/02/17/science/17jeopardy-watson.html

What is IBM Watson?

- Collection of application program interfaces (APIs) that are trained to help process speech, text, and images in a natural way
- 16 products and services in 7 categories

AI Assistant

Data

Knowledge

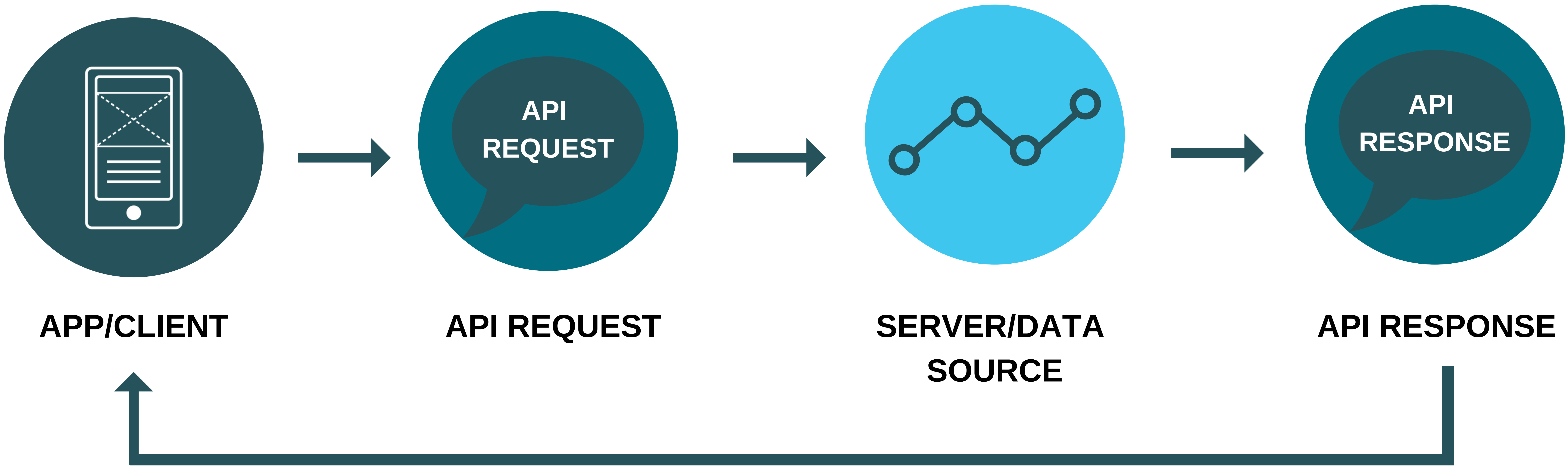
Vision

Speech

Language

Empathy

What is an API?



API
REQUEST

API
RESPONSE

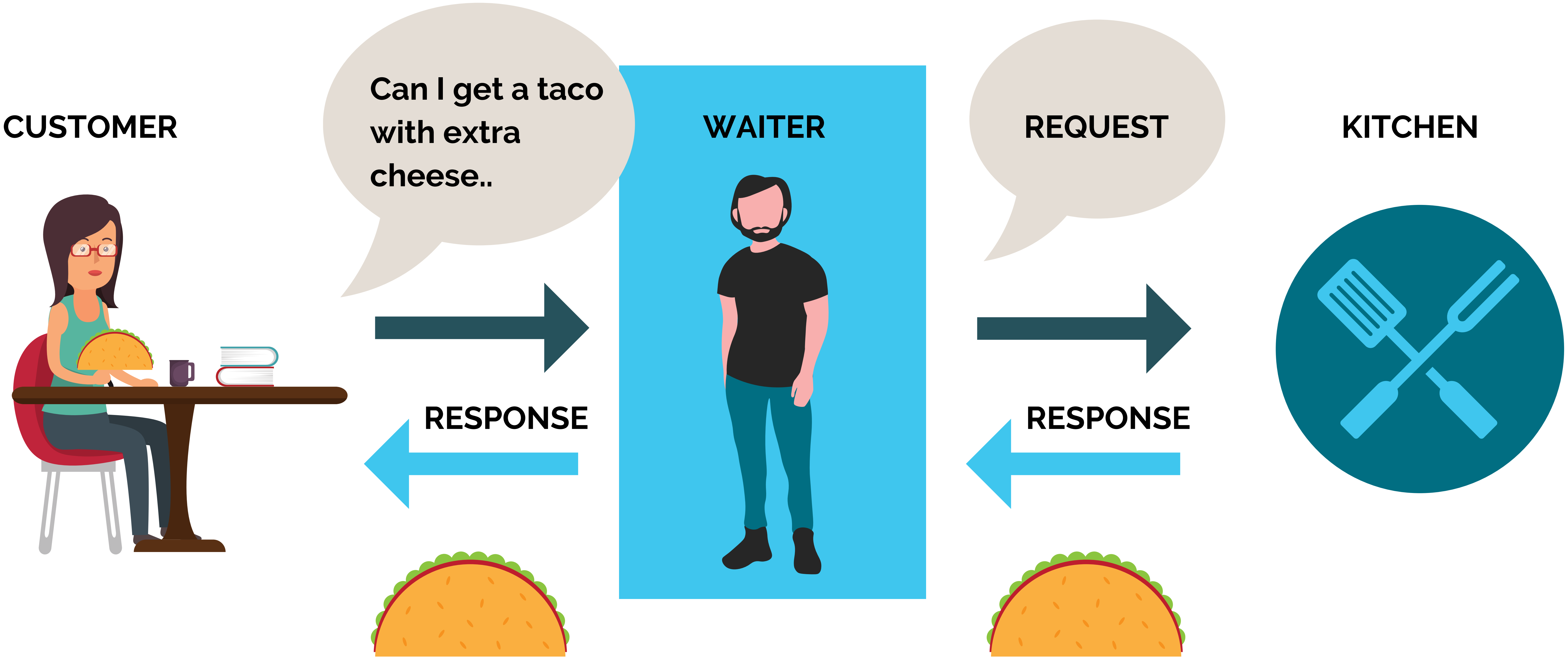
APP/CLIENT

API REQUEST

SERVER/DATA
SOURCE

API RESPONSE

What is an API?

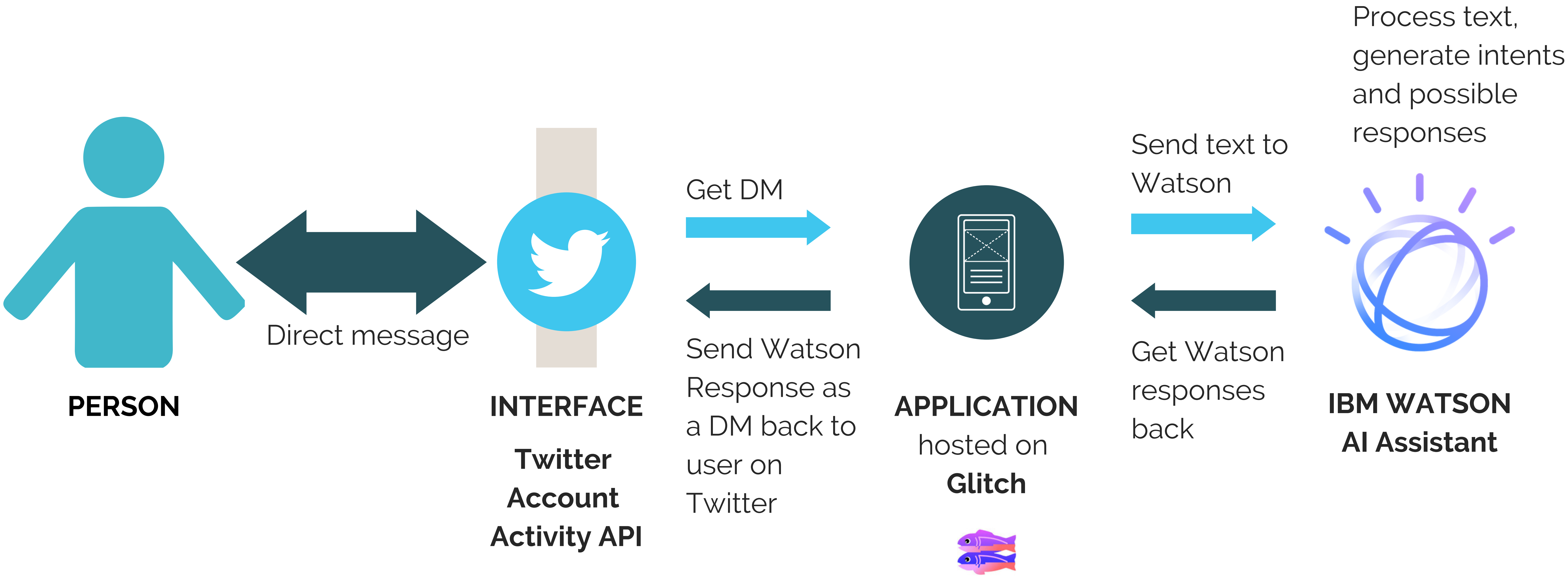


APP/CLIENT

API

DATABASE

Creating a Twitter Chatbot with IBM Watson AI Assistant



Process text, generate intents and possible responses

Direct message

Get DM

Send Watson Response as a DM back to user on Twitter

Send text to Watson

Get Watson responses back

PERSON

INTERFACE

Twitter Account Activity API

APPLICATION hosted on **Glitch**

IBM WATSON AI Assistant

IBM Watson AI Assistant

- Pre-trained with industry relevant content
- Free Lite Plan
  - 10,000 messages/month
  - 5 skills
  - 100 dialog nodes
  - 7 days worth of analytic data

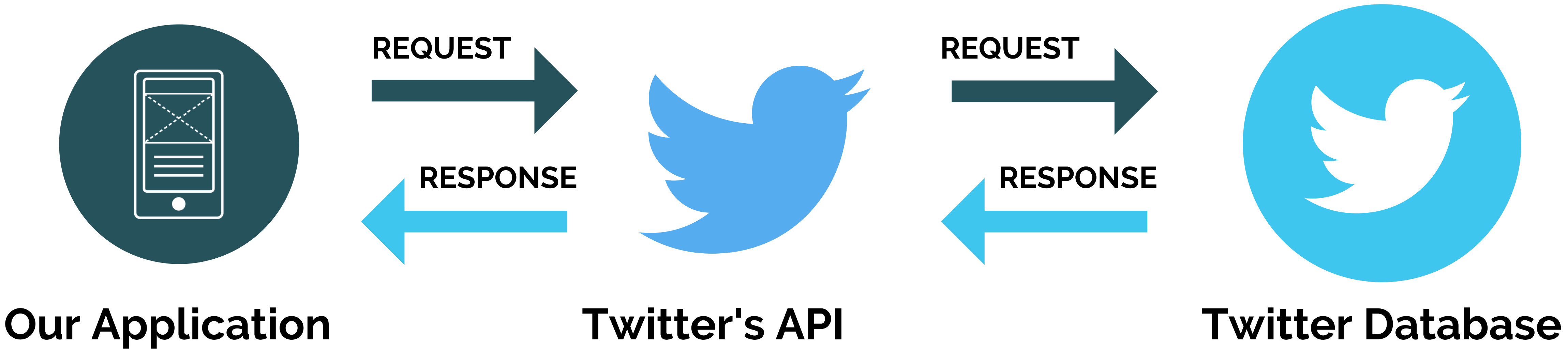
Process text, generate intents and possible responses

IBM WATSON
AI ASSISTANT

Twitter's Account Activity API

- Webhook-based API to send account events

What is a webhook? Consider a scenario where we want to check for a direct message on Twitter



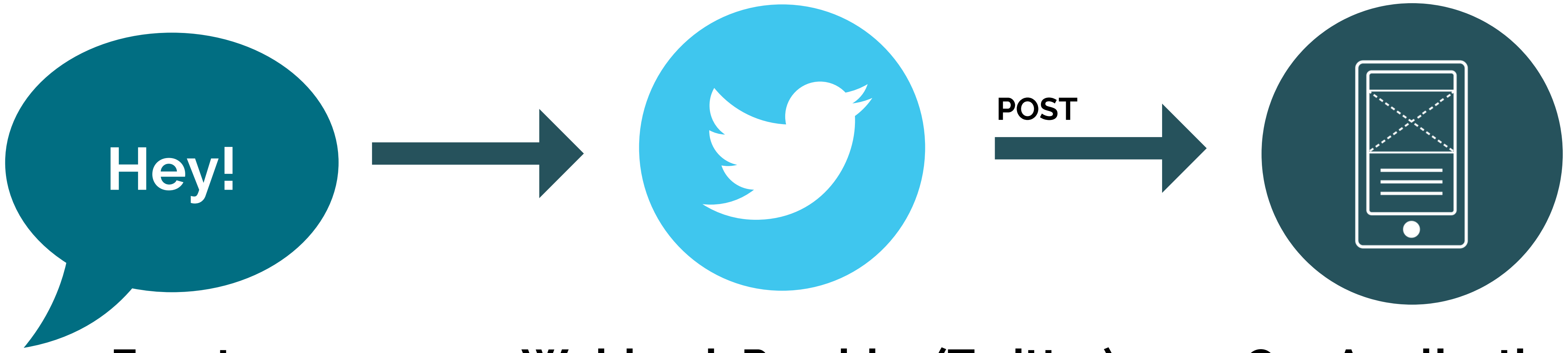
We may have to constantly request until a DM is received...

Do we have a DM yet? Do we have a DM yet? Do we have a DM yet? Do we have a DM yet? Do we have a DM yet? Do we have a DM yet? Do we have a DM yet? Do we have a DM yet? Do we have a DM yet? .....

Twitter's Account Activity API

- Webhook-based API to send account events

What if we had a webhook?



Event

Receive a direct message
on registered user account

Webhook Provider (Twitter)

Webhook triggers callback

Our Application

receive POST request on
yourdomain.com/webhook/twitter

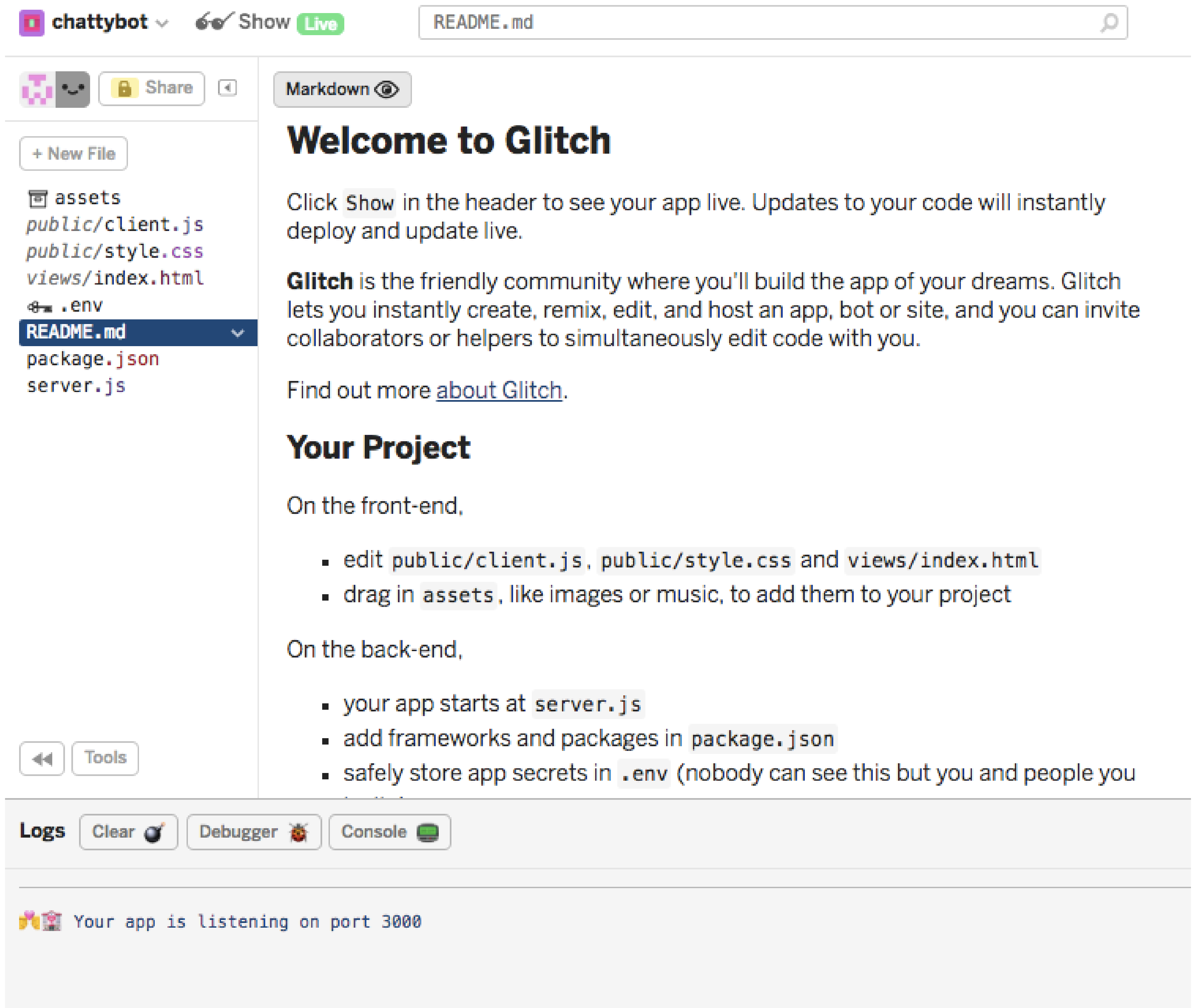
While APIs are request-based, **webhooks are event-based**

Glitch

- Free collaborative platform for creating, remixing, and hosting apps
- Community to find apps and websites

FEATURES

- Instant hosting and automated deployment
  - Instantly get a URL
  - App continuously deployed as you type
  - Preview changes immediately
  - Built-in HTTPS
  - Automatic CDN for assets
- Visual version control
- Server-side debugger
- Embeds
- NPM built-in
- Import Git repos, full console access and more



Welcome to Glitch

Click `Show` in the header to see your app live. Updates to your code will instantly deploy and update live.

Glitch is the friendly community where you'll build the app of your dreams. Glitch lets you instantly create, remix, edit, and host an app, bot or site, and you can invite collaborators or helpers to simultaneously edit code with you.

Find out more about Glitch.

Your Project

On the front-end,

- edit `public/client.js`, `public/style.css` and `views/index.html`
- drag in `assets`, like images or music, to add them to your project

On the back-end,

- your app starts at `server.js`
- add frameworks and packages in `package.json`
- safely store app secrets in `.env` (nobody can see this but you and people you

```
Your app is listening on port 3000
```

Project Part I: Setting up Twitter

1. Apply for a **Twitter developer account.**

2. Login and create a Twitter app and set Access permission to "Read, write, and direct messages."

3. Set up an **Account Activity API Twitter Dev environment**.

4. Create a node application on **Glitch**.

5. Generate access keys and tokens and set up Glitch .env file.

6. Write code that receives a Twitter Challenge Response Check (CRC) GET request. App needs to build an encrypted response_token encoded in JSON.

   See details here.

   See example in Node/JS here.

7. Setup and run the **Account Activity Dashboard application.**

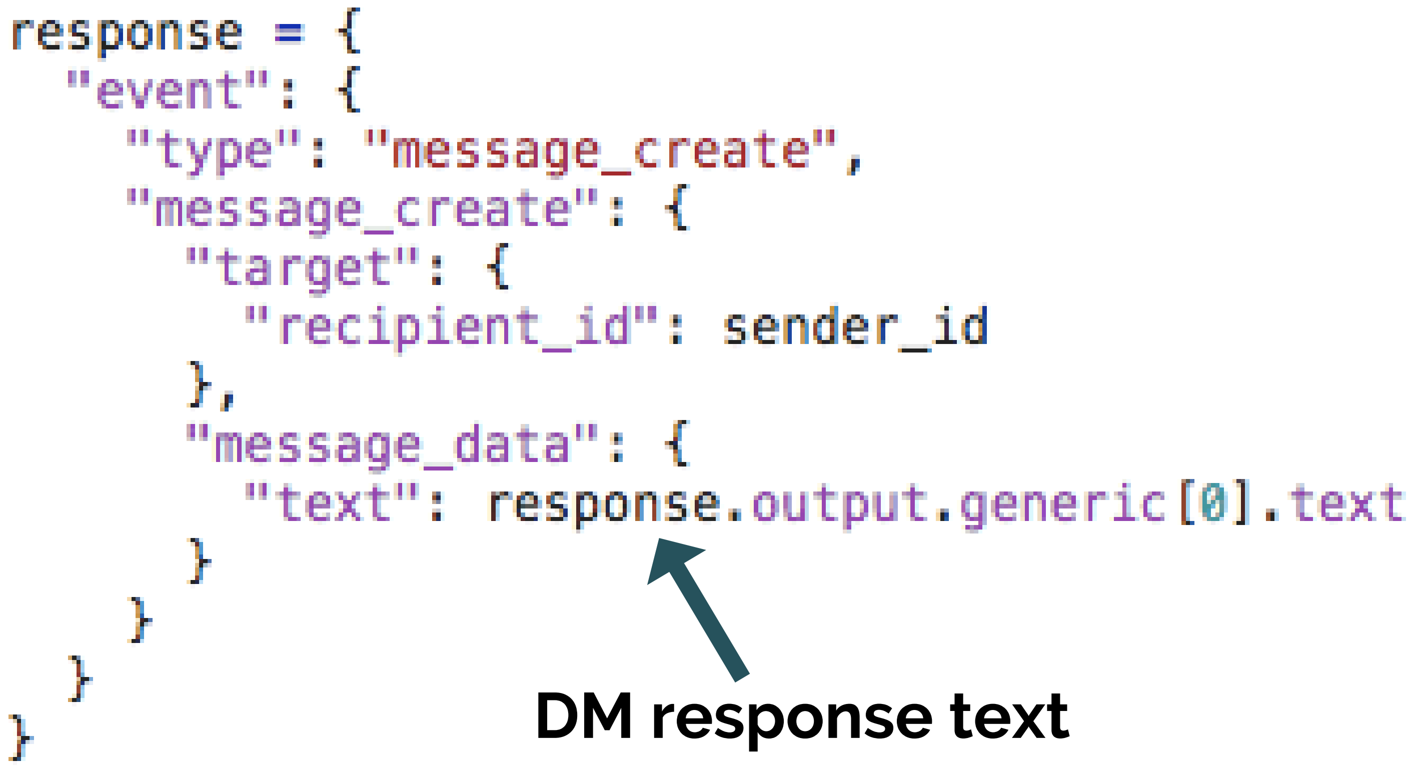
8. Follow the Account Activity Dashboard README.md instructions to configure webhook to receive events and add a user subscription.

9. In your app, create a POST request to your webhook URL and check for direct_message_events

10. For each message event, if the message event type is "message_create" and the sender_id is not your own (**use this to convert username to Twitter id**) then #11.

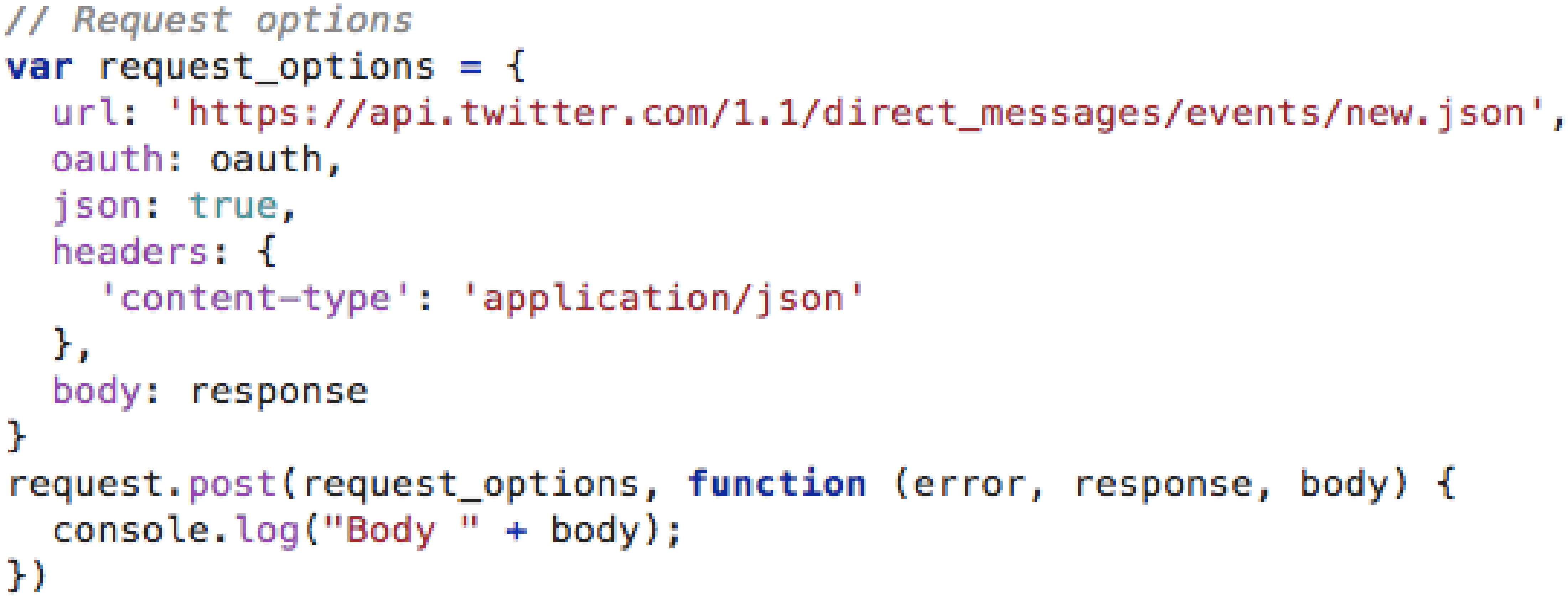
11. Send a POST request to https://api.twitter.com/1.1/direct_messages/events/new.json with the following response. oauth is an object that contains your consumer_key, consumer_secret, token, and token_secret.

12. Send a DM to the account this app is subscribed on. An immediate response should be generated corresponding to what was placed in message_data.text



```
response = {
  "event": {
    "type": "message_create",
    "message_create": {
      "target": {
        "recipient_id": sender_id
      },
      "message_data": {
        "text": response.output.generic[0].text
      }
    }
  }
}
```

DM response text



```
// Request options
var request_options = {
  url: 'https://api.twitter.com/1.1/direct_messages/events/new.json',
  oauth: oauth,
  json: true,
  headers: {
    'content-type': 'application/json'
  },
  body: response
}
request.post(request_options, function (error, response, body) {
  console.log("Body " + body);
})
```

Project Part II: Integrating IBM Watson

1. Create an **IBM Cloud account**.

2. Catalog -> Choose AI category -> Click on Watson Assistant -> Chatbot

3. Click Launch Tool

4. Create a new skill. Explore sample skill to see how intent, entities, and dialog connect.

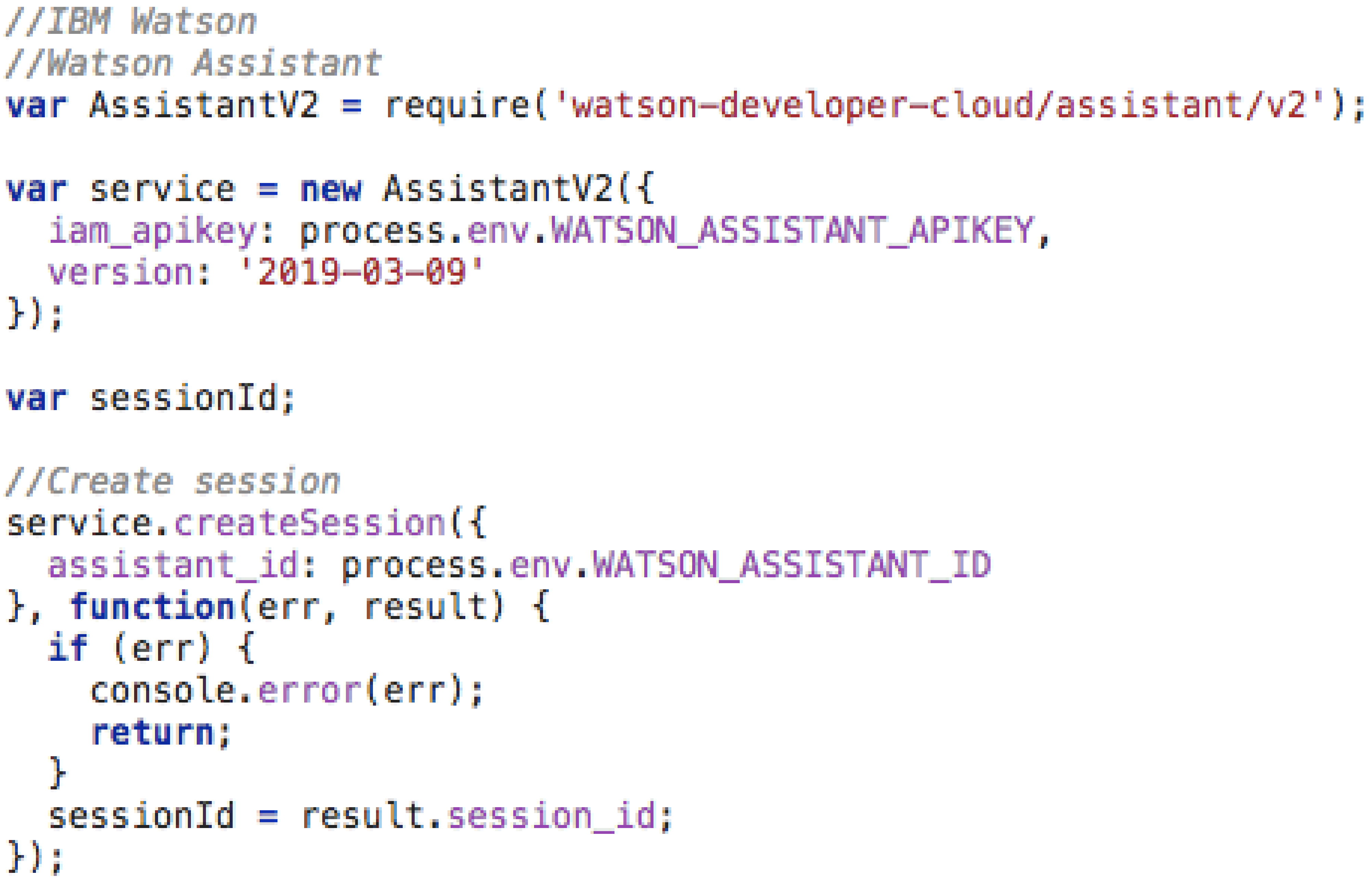
5. Add intents, dialog, entities.

6. Create a new Assistant (click on top Assistant tab next to Skills). Add the dialog skill you created.

7. On the Assistants page, at the right of your new Assistant, click on the series of dots (ellipsis symbol) to the right and View API Details to retrieve the Assistant ID.

8. Setup the correct service endpoint based on location of service. Example below is for Dallas.

9. Create a session. More details can be found on the **Watson Assistant v2 API Docs**.



```
//IBM Watson
//Watson Assistant
var AssistantV2 = require('watson-developer-cloud/assistant/v2');

var service = new AssistantV2({
  iam_apikey: process.env.WATSON_ASSISTANT_APIKEY,
  version: '2019-03-09'
});

var sessionId;

//Create session
service.createSession({
  assistant_id: process.env.WATSON_ASSISTANT_ID
}, function(err, result) {
  if (err) {
    console.error(err);
    return;
  }
  sessionId = result.session_id;
});
```

All code can be found on Glitch here.

Thank You