

How to Build a Quokkabot

@amys_kapers

How to Build a Quokkabot

@amys_kapers

I acknowledge the
traditional owners of this
land, the **Gadigal** people of
the **Eora** nation.

@amys_kapers



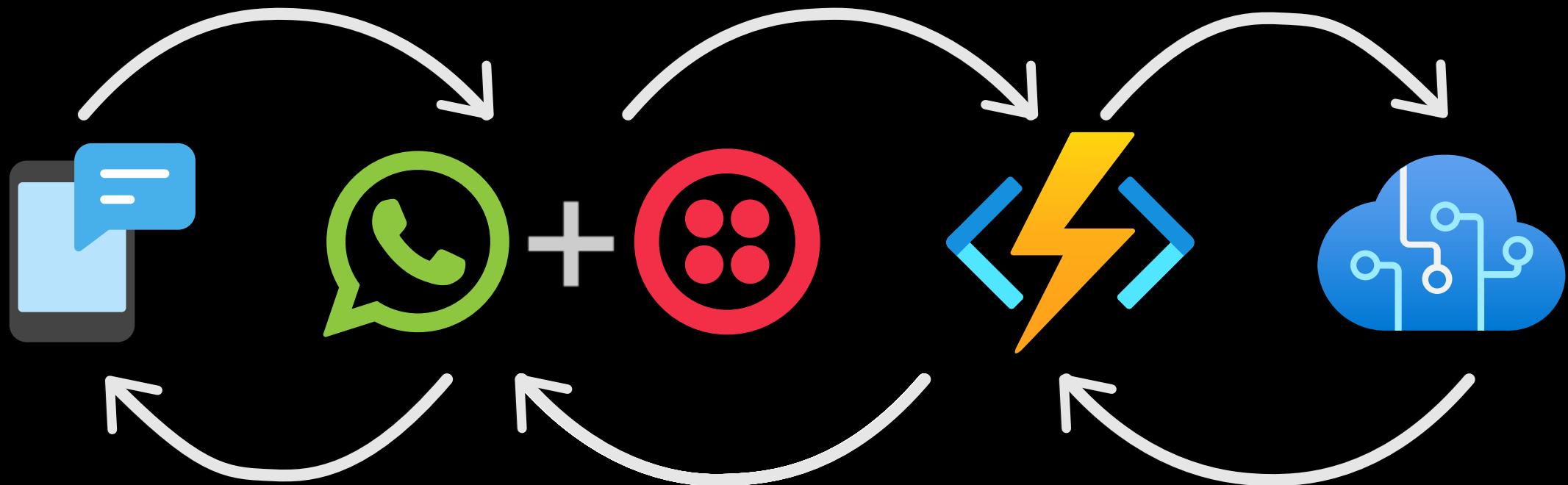
How to Build a Quokkabot

@amys_kapers









Demo Time



@amys_kapers

```
require('dotenv').config()

const twilio = require('twilio')

module.exports = async function (context) {
  const MessagingResponse = twilio.twiml.MessagingResponse
  const twiml = new MessagingResponse()
  const message = twiml.message()

  message.body("Welcome to Quokkabot")

  context.done(null, {
    status: 200,
    body: message.toString(),
    headers: {
      'Content-Type': 'text/xml'
    },
  })
};

};
```

@amys_kapers

```
require('dotenv').config()

const twilio = require('twilio')

module.exports = async function (context) {
    const MessagingResponse = twilio.twiml.MessagingResponse
    const twiml = new MessagingResponse()
    const message = twiml.message()

    message.body("Welcome to Quokkabot")

    context.done(null, {
        status: 200,
        body: message.toString(),
        headers: {
            'Content-Type': 'text/xml'
        },
    })
};

};
```

@amys_kapers

```
require('dotenv').config()

const twilio = require('twilio')

module.exports = async function (context) {
  const MessagingResponse = twilio.twiml.MessagingResponse
  const twiml = new MessagingResponse()
  const message = twiml.message()

  message.body("Welcome to Quokkabot")

  context.done(null, {
    status: 200,
    body: message.toString(),
    headers: {
      'Content-Type': 'text/xml'
    },
  })
};

};
```

@amys_kapers

```
require('dotenv').config()

const twilio = require('twilio')

module.exports = async function (context) {
  const MessagingResponse = twilio.twiml.MessagingResponse
  const twiml = new MessagingResponse()
  const message = twiml.message()

  message.body("Welcome to Quokkabot")

  context.done(null, {
    status: 200,
    body: message.toString(),
    headers: {
      'Content-Type': 'text/xml'
    },
  })
};

};
```

@amys_kapers

```
require('dotenv').config()

const twilio = require('twilio')

module.exports = async function (context) {
  const MessagingResponse = twilio.twiml.MessagingResponse
  const twiml = new MessagingResponse()
  const message = twiml.message()

  message.body("Welcome to Quokkabot")

  context.done(null, {
    status: 200,
    body: message.toString(),
    headers: {
      'Content-Type': 'text/xml'
    },
  })
};

};
```

@amys_kapers

```
require('dotenv').config()

const twilio = require('twilio')

module.exports = async function (context) {
  const MessagingResponse = twilio.twiml.MessagingResponse
  const twiml = new MessagingResponse()
  const message = twiml.message()

  message.body("Welcome to Quokkabot")

  context.done(null, {
    status: 200,
    body: message.toString(),
    headers: {
      'Content-Type': 'text/xml'
    },
  })
};

};
```

```
const twilio = require('twilio')

const { randomImage, quokkas, notQuokkas } = require('../_data/photos')

module.exports = async function (context) {
  const MessagingResponse = twilio.twiml.MessagingResponse
  const twiml = new MessagingResponse()
  const message = twiml.message()
  const msgParams = new URLSearchParams(context.req.body)
  const msgText = msgParams.get('Body')

  if(RegExp('quokka', 'i').test(msgText)) {
    message.body('This is a quokka')
    message.media(
      `https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`
    )
  }
  else {
    message.body('This is not a quokka')
    message.media(
      `https://quokkas.amyskapers.dev/img/not_quokkas/${randomImage(notQuokkas).slug}`
    )
  }

  context.done(null, {
    status: 200,
    body: message.toString(),
    headers: {
      'Content-Type': 'text/xml'
    },
  })
};

};
```

@amys_kapers

```
const twilio = require('twilio')

const { randomImage, quokkas, notQuokkas } = require('../_data/photos')

module.exports = async function (context) {
  const MessagingResponse = twilio.twiml.MessagingResponse
  const twiml = new MessagingResponse()
  const message = twiml.message()
  const msgParams = new URLSearchParams(context.req.body)
  const msgText = msgParams.get('Body')

  if(RegExp('quokka', 'i').test(msgText)) {
    message.body('This is a quokka')
    message.media(
      `https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`
    )
  }
  else {
    message.body('This is not a quokka')
    message.media(
      `https://quokkas.amyskapers.dev/img/not_quokkas/${randomImage(notQuokkas).slug}`
    )
  }

  context.done(null, {
    status: 200,
    body: message.toString(),
    headers: {
      'Content-Type': 'text/xml'
    },
  })
};

};
```

@amys_kapers

```
const twilio = require('twilio')

const { randomImage, quokkas, notQuokkas } = require('../_data/photos')

module.exports = async function (context) {
  const MessagingResponse = twilio.twiml.MessagingResponse
  const twiml = new MessagingResponse()
  const message = twiml.message()
  const msgParams = new URLSearchParams(context.req.body)
  const msgText = msgParams.get('Body')

  if(RegExp('quokka', 'i').test(msgText)) {
    message.body('This is a quokka')
    message.media(
      `https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`
    )
  }
  else {
    message.body('This is not a quokka')
    message.media(
      `https://quokkas.amyskapers.dev/img/not_quokkas/${randomImage(notQuokkas).slug}`
    )
  }

  context.done(null, {
    status: 200,
    body: message.toString(),
    headers: {
      'Content-Type': 'text/xml'
    },
  })
};

};
```

@amys_kapers

```
const twilio = require('twilio')

const { randomImage, quokkas, notQuokkas } = require('../_data/photos')

module.exports = async function (context) {
  const MessagingResponse = twilio.twiml.MessagingResponse
  const twiml = new MessagingResponse()
  const message = twiml.message()
  const msgParams = new URLSearchParams(context.req.body)
  const msgText = msgParams.get('Body')

  if(RegExp('quokka', 'i').test(msgText)) {
    message.body('This is a quokka')
    message.media(
      `https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`
    )
  }
  else {
    message.body('This is not a quokka')
    message.media(
      `https://quokkas.amyskapers.dev/img/not_quokkas/${randomImage(notQuokkas).slug}`
    )
  }

  context.done(null, {
    status: 200,
    body: message.toString(),
    headers: {
      'Content-Type': 'text/xml'
    },
  })
};

};
```

@amys_kapers

```
const twilio = require('twilio')

const { randomImage, quokkas, notQuokkas } = require('../_data/photos')

module.exports = async function (context) {
  const MessagingResponse = twilio.twiml.MessagingResponse
  const twiml = new MessagingResponse()
  const message = twiml.message()
  const msgParams = new URLSearchParams(context.req.body)
  const msgText = msgParams.get('Body')

  if(RegExp('quokka', 'i').test(msgText)) {
    message.body('This is a quokka')
    message.media(
      `https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`
    )
  }
  else {
    message.body('This is not a quokka')
    message.media(
      `https://quokkas.amyskapers.dev/img/not_quokkas/${randomImage(notQuokkas).slug}`
    )
  }

  context.done(null, {
    status: 200,
    body: message.toString(),
    headers: {
      'Content-Type': 'text/xml'
    },
  })
};

};
```

@amys_kapers

Try it yourself!

@amys_kapers



```
// Existing imports here

const { PredictionAPIClient } = require('@azure/cognitiveservices-customvision-prediction')
const { ApiKeyCredentials } = require('@azure/ms-rest-js')

const key = process.env.API_KEY
const endpoint = process.env.ENDPOINT
const projectId = process.env.PROJECT_ID
const iteration = process.env.ITERATION

const credentials = new ApiKeyCredentials({ inHeader: { "Prediction-key": key } })
const predictor = new PredictionAPIClient(credentials, endpoint)

module.exports = async function (context) {
    // Existing code here

    const image = msgParams.get('MediaUrl0')
    let results = false

    if(image) {
        results = await predictor.classifyImageUrl(projectId, iteration, { url: image })
    }
    else {
        if(RegExp('quokka', 'i').test(msgText)) {
            message.body('This is a quokka')
            message.media(`https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`)
        }
        else {
            message.body('This is not a quokka')
            message.media(`https://quokkas.amyskapers.dev/img/not_quokkas/${randomImage(notQuokkas).slug}`)
        }
    }

    context.log({results: JSON.stringify(results)})
    // Finish Azure function
};
```

@amys_kapers

```

// Existing imports here

const { PredictionAPIClient } = require('@azure/cognitiveservices-customvision-prediction')
const { ApiKeyCredentials } = require('@azure/ms-rest-js')

const key = process.env.API_KEY
const endpoint = process.env.ENDPOINT
const projectId = process.env.PROJECT_ID
const iteration = process.env.ITERATION

const credentials = new ApiKeyCredentials({ inHeader: { "Prediction-key": key } })
const predictor = new PredictionAPIClient(credentials, endpoint)

module.exports = async function (context) {
    // Existing code here

    const image = msgParams.get('MediaUrl0')
    let results = false

    if(image) {
        results = await predictor.classifyImageUrl(projectId, iteration, { url: image })
    }
    else {
        if(RegExp('quokka', 'i').test(msgText)) {
            message.body('This is a quokka')
            message.media(`https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`)
        }
        else {
            message.body('This is not a quokka')
            message.media(`https://quokkas.amyskapers.dev/img/not_quokkas/${randomImage(notQuokkas).slug}`)
        }
    }

    context.log({results: JSON.stringify(results)})
}

// Finish Azure function
};
```

@amys_kapers

```

// Existing imports here

const { PredictionAPIClient } = require('@azure/cognitiveservices-customvision-prediction')
const { ApiKeyCredentials } = require('@azure/ms-rest-js')

const key = process.env.API_KEY
const endpoint = process.env.ENDPOINT
const projectId = process.env.PROJECT_ID
const iteration = process.env.ITERATION

const credentials = new ApiKeyCredentials({ inHeader: { "Prediction-key": key } })
const predictor = new PredictionAPIClient(credentials, endpoint)

module.exports = async function (context) {
    // Existing code here

    const image = msgParams.get('MediaUrl0')
    let results = false

    if(image) {
        results = await predictor.classifyImageUrl(projectId, iteration, { url: image })
    }
    else {
        if(RegExp('quokka', 'i').test(msgText)) {
            message.body('This is a quokka')
            message.media(`https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`)
        }
        else {
            message.body('This is not a quokka')
            message.media(`https://quokkas.amyskapers.dev/img/not_quokkas/${randomImage(notQuokkas).slug}`)
        }
    }

    context.log({results: JSON.stringify(results)})
};

// Finish Azure function
};

```

@amys_kapers

```
// Existing imports here

const { PredictionAPIClient } = require('@azure/cognitiveservices-customvision-prediction')
const { ApiKeyCredentials } = require('@azure/ms-rest-js')

const key = process.env.API_KEY
const endpoint = process.env.ENDPOINT
const projectId = process.env.PROJECT_ID
const iteration = process.env.ITERATION

const credentials = new ApiKeyCredentials({ inHeader: { "Prediction-key": key } })
const predictor = new PredictionAPIClient(credentials, endpoint)

module.exports = async function (context) {
    // Existing code here

    const image = msgParams.get('MediaUrl0')
    let results = false

    if(image) {
        results = await predictor.classifyImageUrl(projectId, iteration, { url: image })
    }
    else {
        if(RegExp('quokka', 'i').test(msgText)) {
            message.body('This is a quokka')
            message.media(`https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`)
        }
        else {
            message.body('This is not a quokka')
            message.media(`https://quokkas.amyskapers.dev/img/not_quokkas/${randomImage(notQuokkas).slug}`)
        }
    }
}

context.log({results: JSON.stringify(results)})

// Finish Azure function
};
```

@amys_kapers

```

// Existing imports here

const { PredictionAPIClient } = require('@azure/cognitiveservices-customvision-prediction')
const { ApiKeyCredentials } = require('@azure/ms-rest-js')

const key = process.env.API_KEY
const endpoint = process.env.ENDPOINT
const projectId = process.env.PROJECT_ID
const iteration = process.env.ITERATION

const credentials = new ApiKeyCredentials({ inHeader: { "Prediction-key": key } })
const predictor = new PredictionAPIClient(credentials, endpoint)

module.exports = async function (context) {
    // Existing code here

    const image = msgParams.get('MediaUrl0')
    let results = false

    if(image) {
        results = await predictor.classifyImageUrl(projectId, iteration, { url: image })
    }
    else {
        if(RegExp('quokka', 'i').test(msgText)) {
            message.body('This is a quokka')
            message.media(`https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`)
        }
        else {
            message.body('This is not a quokka')
            message.media(`https://quokkas.amyskapers.dev/img/not_quokkas/${randomImage(notQuokkas).slug}`)
        }
    }

    context.log({results: JSON.stringify(results)})
}

// Finish Azure function
};

```

@amys_kapers

```
{  
  "results": {  
    "id": "06f7250b-ffb3-47a1-9508-c928b8c54b87",  
    "project": "a44bf78b-da55-4a95-bfdc-21ac780d2518",  
    "iteration": "8a75c738-d658-4f63-8a2e-e18318f418f8",  
    "created": "2022-10-12T06:31:47.662Z",  
    "predictions": [  
      {  
        "probability": 0.99992716,  
        "tagId": "1f6223d5-79ce-4e6b-8f58-ddfbb4635246",  
        "tagName": "Quokka",  
        "tagType": "Regular"  
      },  
      {  
        "probability": 0.00007281252,  
        "tagId": "7a44e503-b236-4991-bf44-2e08ca0e2356",  
        "tagName": "Negative",  
        "tagType": "Negative"  
      }  
    ]  
  }  
}
```

```
"predictions": [  
    {  
        "probability": 0.99992716,  
        "tagId": "1f6223d5-79ce-4e6b-8f58-ddfb4635246",  
        "tagName": "Quokka",  
        "tagType": "Regular"  
    },  
    {  
        "probability": 0.00007281252,  
        "tagId": "7a44e503-b236-4991-bf44-2e08ca0e2356",  
        "tagName": "Negative",  
        "tagType": "Negative"  
    }  
]
```

```
"predictions": [  
    {  
        "probability": 0.99992716,  
        "tagName": "Quokka",  
    },  
    {  
        "probability": 0.00007281252,  
        "tagName": "Negative",  
    }  
]
```

```
// Existing imports here

module.exports = async function (context) {
    // Existing code here

    if(image) {
        // Get image classification results

        let outcome = {}

        results.predictions.forEach(tag => {
            if (tag.tagName == 'Negative') {
                outcome.negative = tag.probability
            } else if (tag.tagName == 'Quokka') {
                outcome.quokka = tag.probability
            }
        })

        const quokka = `${(outcome.quokka * 100).toFixed(2)}%`
        const notQuokka = `${(outcome.negative * 100).toFixed(2)}%`

        if(outcome.negative > outcome.quokka) {
            message.body(`Sorry that doesn't look like a quokka\nQuokka: ${quokka}, Not Quokka: ${notQuokka}\nThat's pretty sad though, so here's a quokka`)
        }

        message.media(`https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`)
    } else {
        message.body(`Yep that looks like a quokka!\nQuokka: ${quokka}, Not Quokka: ${notQuokka}`)
    }
} else {
    // Send Quokka or not Quokka photo
}

// Finish Azure Function
};
```

@amys_kapers

```
// Existing imports here

module.exports = async function (context) {
    // Existing code here

    if(image) {
        // Get image classification results

        let outcome = {}

        results.predictions.forEach(tag => {
            if (tag.tagName == 'Negative') {
                outcome.negative = tag.probability
            } else if (tag.tagName == 'Quokka') {
                outcome.quokka = tag.probability
            }
        })

        const quokka = `${(outcome.quokka * 100).toFixed(2)}%`
        const notQuokka = `${(outcome.negative * 100).toFixed(2)}%`

        if(outcome.negative > outcome.quokka) {
            message.body(`Sorry that doesn't look like a quokka\nQuokka: ${quokka}, Not Quokka: ${notQuokka}\nThat's pretty sad though, so here's a quokka`)
            message.media(`https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`)
        } else {
            message.body(`Yep that looks like a quokka!\nQuokka: ${quokka}, Not Quokka: ${notQuokka}`)
        }
    } else {
        // Send Quokka or not Quokka photo
    }
}

// Finish Azure Function
};
```

@amys_kapers

```
// Existing imports here

module.exports = async function (context) {
    // Existing code here

    if(image) {
        // Get image classification results

        let outcome = {}

        results.predictions.forEach(tag => {
            if (tag.tagName == 'Negative') {
                outcome.negative = tag.probability
            } else if (tag.tagName == 'Quokka') {
                outcome.quokka = tag.probability
            }
        })

        const quokka = ` ${(outcome.quokka * 100).toFixed(2)}%`
        const notQuokka = `${(outcome.negative * 100).toFixed(2)}%`

        if(outcome.negative > outcome.quokka) {
            message.body(`Sorry that doesn't look like a quokka\nQuokka: ${quokka}, Not Quokka: ${notQuokka}\nThat's pretty sad though, so here's a quokka`)
        }

        message.media(`https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`)
    } else {
        message.body(`Yep that looks like a quokka!\nQuokka: ${quokka}, Not Quokka: ${notQuokka}`)
    }
} else {
    // Send Quokka or not Quokka photo
}

// Finish Azure Function
};
```

@amys_kapers

```
// Existing imports here

module.exports = async function (context) {
    // Existing code here

    if(image) {
        // Get image classification results

        let outcome = {}

        results.predictions.forEach(tag => {
            if (tag.tagName == 'Negative') {
                outcome.negative = tag.probability
            } else if (tag.tagName == 'Quokka') {
                outcome.quokka = tag.probability
            }
        })

        const quokka = `${(outcome.quokka * 100).toFixed(2)}%`
        const notQuokka = `${(outcome.negative * 100).toFixed(2)}%`

        if(outcome.negative > outcome.quokka) {
            message.body(`Sorry that doesn't look like a quokka\nQuokka: ${quokka}, Not Quokka: ${notQuokka}\nThat's pretty sad though, so here's a quokka`)
        }

        message.media(`https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`)
    } else {
        message.body(`Yep that looks like a quokka!\nQuokka: ${quokka}, Not Quokka: ${notQuokka}`)
    }
} else {
    // Send Quokka or not Quokka photo
}

// Finish Azure Function
};
```

@amys_kapers

Try it yourself!

@amys_kapers



```
// Existing imports here

module.exports = async function (context) {
    // Existing code here

    if(image) {
        // Check Image
    }
    else {
        if(RegExp(/error|issue|wrong/, 'i').test(msgText)) {
            message.body('Sorry about that, here is a quokka to cheer you up')
            message.media(
                `https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`
            )
        }
        else if(RegExp('quokka', 'i').test(msgText)) {
            // Send Quokka
        }
        else {
            // Send not Quokka
        }
    }

    // Finish Azure Function
};


```

@amys_kapers

```
// Existing imports here

module.exports = async function (context) {
    // Existing code here

    if(image) {
        // Check Image
    }
    else {
        if(RegExp(/error|issue|wrong/, 'i').test(msgText)) {
            message.body('Sorry about that, here is a quokka to cheer you up')
            message.media(
                `https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`
            )
        }
        else if(RegExp('quokka', 'i').test(msgText)) {
            // Send Quokka
        }
        else {
            // Send not Quokka
        }
    }

    // Finish Azure Function
};


```

@amys_kapers

```
// Existing imports here

module.exports = async function (context) {
    // Existing code here

    if(image) {
        // Check Image
    }
    else {
        if(RegExp(/error|issue|wrong/, 'i').test(msgText)) {
            message.body('Sorry about that, here is a quokka to cheer you up')
            message.media(
                `https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`
            )
        }
        else if(RegExp('quokka', 'i').test(msgText)) {
            // Send Quokka
        }
        else {
            // Send not Quokka
        }
    }

    // Finish Azure Function
};


```

@amys_kapers

```
// Existing imports here

module.exports = async function (context) {
    // Existing code here

    if(image) {
        // Check Image
    }
    else {
        if(RegExp(/error|issue|wrong/, 'i').test(msgText)) {
            message.body('Sorry about that, here is a quokka to cheer you up')
            message.media(
                `https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`
            )
        }
        else if(RegExp('quokka', 'i').test(msgText)) {
            // Send Quokka
        }
        else {
            // Send not Quokka
        }
    }

    // Finish Azure Function
};


```

@amys_kapers

```
// Existing imports here

module.exports = async function (context) {
    // Existing code here

    if(image) {
        // Check Image
    }
    else {
        if(RegExp(/error|issue|wrong/, 'i').test(msgText)) {
            message.body('Sorry about that, here is a quokka to cheer you up')
        }
        else if (RegExp('fact', 'i').test(msgText)) {
            message.body(randomFacts(facts))
        }
        else if(RegExp('quokka', 'i').test(msgText)) {
            message.body('This is a quokka')

        }
        else {
            message.body(`Welcome to Quokkabot! I can do a bunch of different things that have to
do with quokkas.\nNeed a picture of a quokka? Just ask me\nNot sure if you've seen a quokka? Send
me a picture and I'll tell you if there's a quokka in it`)
        }

        message.media(`https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`)
    }
    // Finish Azure Function
};
```

@amys_kapers

```
// Existing imports here

module.exports = async function (context) {
    // Existing code here

    if(image) {
        // Check Image
    }
    else {
        if(RegExp(/error|issue|wrong/, 'i').test(msgText)) {
            message.body('Sorry about that, here is a quokka to cheer you up')
        }
        else if (RegExp('fact', 'i').test(msgText)) {
            message.body(randomFacts(facts))
        }
        else if(RegExp('quokka', 'i').test(msgText)) {
            message.body('This is a quokka')

        }
        else {
            message.body(`Welcome to Quokkabot! I can do a bunch of different things that have to
do with quokkas.\nNeed a picture of a quokka? Just ask me\nNot sure if you've seen a quokka? Send
me a picture and I'll tell you if there's a quokka in it`)
        }

        message.media(`https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`)
    }
    // Finish Azure Function
};
```

@amys_kapers

```
// Existing imports here

module.exports = async function (context) {
    // Existing code here

    if(image) {
        // Check Image
    }
    else {
        if(RegExp(/error|issue|wrong/, 'i').test(msgText)) {
            message.body('Sorry about that, here is a quokka to cheer you up')
        }
        else if (RegExp('fact', 'i').test(msgText)) {
            message.body(randomFacts(facts))
        }
        else if(RegExp('quokka', 'i').test(msgText)) {
            message.body('This is a quokka')
        }
        else {
            message.body(`Welcome to Quokkabot! I can do a bunch of different things that have to
do with quokkas.\nNeed a picture of a quokka? Just ask me\nNot sure if you've seen a quokka? Send
me a picture and I'll tell you if there's a quokka in it`)
        }
    }

    message.media(`https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`)
}

// Finish Azure Function
};
```

@amys_kapers

```
// Existing imports here

module.exports = async function (context) {
    // Existing code here

    if(image) {
        // Check Image
    }
    else {
        if(RegExp(/error|issue|wrong/, 'i').test(msgText)) {
            message.body('Sorry about that, here is a quokka to cheer you up')
        }
        else if (RegExp('fact', 'i').test(msgText)) {
            message.body(randomFacts(facts))
        }
        else if(RegExp('quokka', 'i').test(msgText)) {
            message.body('This is a quokka')

        }
        else {
            message.body(`Welcome to Quokkabot! I can do a bunch of different things that have to
do with quokkas.\nNeed a picture of a quokka? Just ask me\nNot sure if you've seen a quokka? Send
me a picture and I'll tell you if there's a quokka in it`)
        }

        message.media(`https://quokkas.amyskapers.dev/img/quokkas/${randomImage(quokkas).slug}`)
    }
    // Finish Azure Function
};
```

@amys_kapers

Demo Time



@amys_kapers

```
const sgMail = require('@sendgrid/mail')
const parseMultipartFormData = require('@anzp/azure-function-multipart').default

module.exports = async function (context, req) {
    const { fields, files } = await parseMultipartFormData(req)
    const body = {}
    const image = (files && files.length) && files[0].bufferFile

    fields.forEach(field => {
        body[field.name] = field.value
    })

    sgMail.setApiKey(process.env.SENDGRID_API_KEY)

    let msg = {
        to: body.from,
        from: {
            name: 'Quokkabot',
            email: 'quokkas@amyskapers.dev',
        },
        subject: `Re: ${body.subject}`
    }

    if (image) {
        const results = await predictor.classifyImage(projectId, iteration, image)
        // Format and return image results

        msg.html = formatResults(results)
    } else {
        // Send a Quokka, fact or information
    }

    msg.html = `${msg.html}`

    await sgMail.send(msg)
};


```

@amys_kapers

```
const sgMail = require('@sendgrid/mail')
const parseMultipartFormData = require('@anzp/azure-function-multipart').default

module.exports = async function (context, req) {
    const { fields, files } = await parseMultipartFormData(req)
    const body = {}
    const image = (files && files.length) && files[0].bufferFile

    fields.forEach(field => {
        body[field.name] = field.value
    })

    sgMail.setApiKey(process.env.SENDGRID_API_KEY)

    let msg = {
        to: body.from,
        from: {
            name: 'Quokkabot',
            email: 'quokkas@amyskapers.dev',
        },
        subject: `Re: ${body.subject}`
    }

    if (image) {
        const results = await predictor.classifyImage(projectId, iteration, image)
        // Format and return image results

        msg.html = formatResults(results)
    } else {
        // Send a Quokka, fact or information
    }

    msg.html = `${msg.html}`

    await sgMail.send(msg)
};


```

@amys_kapers

```
const sgMail = require('@sendgrid/mail')
const parseMultipartFormData = require('@anzp/azure-function-multipart').default

module.exports = async function (context, req) {
  const { fields, files } = await parseMultipartFormData(req)
  const body = {}
  const image = (files && files.length) && files[0].bufferFile

  fields.forEach(field => {
    body[field.name] = field.value
  })

  sgMail.setApiKey(process.env.SENDGRID_API_KEY)

  let msg = {
    to: body.from,
    from: {
      name: 'Quokkabot',
      email: 'quokkas@amyskapers.dev',
    },
    subject: `Re: ${body.subject}`
  }

  if (image) {
    const results = await predictor.classifyImage(projectId, iteration, image)
    // Format and return image results

    msg.html = formatResults(results)
  } else {
    // Send a Quokka, fact or information
  }

  msg.html = `${msg.html}`

  await sgMail.send(msg)
};


```

@amys_kapers

```
const sgMail = require('@sendgrid/mail')
const parseMultipartFormData = require('@anzp/azure-function-multipart').default

module.exports = async function (context, req) {
  const { fields, files } = await parseMultipartFormData(req)
  const body = {}
  const image = (files && files.length) && files[0].bufferFile

  fields.forEach(field => {
    body[field.name] = field.value
  })

  sgMail.setApiKey(process.env.SENDGRID_API_KEY)

  let msg = {
    to: body.from,
    from: {
      name: 'Quokkabot',
      email: 'quokkas@amyskapers.dev',
    },
    subject: `Re: ${body.subject}`
  }

  if (image) {
    const results = await predictor.classifyImage(projectId, iteration, image)
    // Format and return image results

    msg.html = formatResults(results)
  } else {
    // Send a Quokka, fact or information
  }

  msg.html = `${msg.html}`

  await sgMail.send(msg)
};


```

@amys_kapers

```
const sgMail = require('@sendgrid/mail')
const parseMultipartFormData = require('@anzp/azure-function-multipart').default

module.exports = async function (context, req) {
    const { fields, files } = await parseMultipartFormData(req)
    const body = {}
    const image = (files && files.length) && files[0].bufferFile

    fields.forEach(field => {
        body[field.name] = field.value
    })

    sgMail.setApiKey(process.env.SENDGRID_API_KEY)

    let msg = {
        to: body.from,
        from: {
            name: 'Quokkabot',
            email: 'quokkas@amyskapers.dev',
        },
        subject: `Re: ${body.subject}`
    }

    if (image) {
        const results = await predictor.classifyImage(projectId, iteration, image)
        // Format and return image results

        msg.html = formatResults(results)
    } else {
        // Send a Quokka, fact or information
    }

    msg.html = `${msg.html}`

    await sgMail.send(msg)
};


```

@amys_kapers

```
const sgMail = require('@sendgrid/mail')
const parseMultipartFormData = require('@anzp/azure-function-multipart').default

module.exports = async function (context, req) {
    const { fields, files } = await parseMultipartFormData(req)
    const body = {}
    const image = (files && files.length) && files[0].bufferFile

    fields.forEach(field => {
        body[field.name] = field.value
    })

    sgMail.setApiKey(process.env.SENDGRID_API_KEY)

    let msg = {
        to: body.from,
        from: {
            name: 'Quokkabot',
            email: 'quokkas@amyskapers.dev',
        },
        subject: `Re: ${body.subject}`
    }

    if (image) {
        const results = await predictor.classifyImage(projectId, iteration, image)
        // Format and return image results

        msg.html = formatResults(results)
    }
    else {
        // Send a Quokka, fact or information
    }

    msg.html = `${msg.html}`

    await sgMail.send(msg)
};


```

@amys_kapers

```
const sgMail = require('@sendgrid/mail')
const parseMultipartFormData = require('@anzp/azure-function-multipart').default

module.exports = async function (context, req) {
    const { fields, files } = await parseMultipartFormData(req)
    const body = {}
    const image = (files && files.length) && files[0].bufferFile

    fields.forEach(field => {
        body[field.name] = field.value
    })

    sgMail.setApiKey(process.env.SENDGRID_API_KEY)

    let msg = {
        to: body.from,
        from: {
            name: 'Quokkabot',
            email: 'quokkas@amyskapers.dev',
        },
        subject: `Re: ${body.subject}`
    }

    if (image) {
        const results = await predictor.classifyImage(projectId, iteration, image)
        // Format and return image results

        msg.html = formatResults(results)
    }
    else {
        // Send a Quokka, fact or information
    }

    msg.html = `${msg.html}`

    await sgMail.send(msg)
};


```

@amys_kapers

```
const sgMail = require('@sendgrid/mail')
const parseMultipartFormData = require('@anzp/azure-function-multipart').default

module.exports = async function (context, req) {
    const { fields, files } = await parseMultipartFormData(req)
    const body = {}
    const image = (files && files.length) && files[0].bufferFile

    fields.forEach(field => {
        body[field.name] = field.value
    })

    sgMail.setApiKey(process.env.SENDGRID_API_KEY)

    let msg = {
        to: body.from,
        from: {
            name: 'Quokkabot',
            email: 'quokkas@amyskapers.dev',
        },
        subject: `Re: ${body.subject}`
    }

    if (image) {
        const results = await predictor.classifyImage(projectId, iteration, image)
        // Format and return image results

        msg.html = formatResults(results)
    } else {
        // Send a Quokka, fact or information
    }

    msg.html = `${msg.html}`

    await sgMail.send(msg)
};


```

@amys_kapers

Try it yourself!

@amys_kapers

hi@quokkas.dev

@amys_kapers

quokkas.amyskapers.dev

@amys_kapers

quokkas.amyskapers.dev/results

@amys_kapers

What next?

@amys_kapers

Code

Issues 1

Pull requests

Actions

Projects

Security

Insights

prod ▾

2 branches

0 tags

Go to file

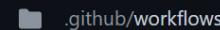
Code ▾



amykapernick trigger build

107b56a 20 hours ago

123 commits



.github/workflows

updated azure deploy function

4 months ago



.vscode

have function working locally

2 years ago



api

fixed image

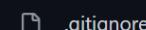
4 months ago



site

bot working

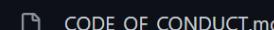
4 months ago



.gitignore

added content

3 years ago



CODE_OF_CONDUCT.md

added contributing guides and license

2 years ago



CONTRIBUTING.md

added contributing guides and license

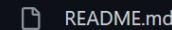
2 years ago



LICENSE

added contributing guides and license

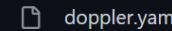
2 years ago



README.md

trigger build

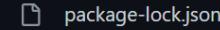
20 hours ago



doppler.yaml

seeing if I broke things

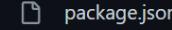
4 months ago



package-lock.json

seeing if I broke things

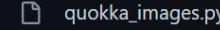
4 months ago



package.json

seeing if I broke things

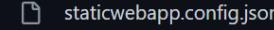
4 months ago



quokka_images.py

updated quokkabot images

4 months ago



staticwebapp.config.json

added config with runtime

4 months ago



tsconfig.json

switched to astro and added new images

4 months ago

About

[quokkas.amyskapers.dev](#)

hacktoberfest

[Readme](#)[MIT license](#)[Code of conduct](#)[1 star](#)[1 watching](#)[12 forks](#)

Releases

No releases published

Packages

No packages published

Contributors 12

github.com/amykapernick/quokkas

@amys_kapers

[Code](#) [Issues 2](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#) is:issue is:open

Labels 13

Milestones 0

New issue

2 Open ✓ 1 Closed

Author ▾ Label ▾ Projects ▾ Milestones ▾ Assignee ▾ Sort ▾

[Add Twitter Version](#) hacktoberfest

#49 opened now by amykapernick

3

[Quokka Facts](#) Always Open good first issue hacktoberfest

#5 opened on 1 Oct 2020 by amykapernick

 **ProTip!** Follow long discussions with [comments:>50](#).

© 2022 GitHub, Inc.

[Terms](#)[Privacy](#)[Security](#)[Status](#)[Docs](#)[Contact GitHub](#)[Pricing](#)[API](#)[Training](#)[Blog](#)[About](#)github.com/amykapernick/quokkas

@amys_kapers

But why?

@amys_kapers

Thank You 

@amys_kapers

Questions?

@amys_kapers