

QConSF

Chatbots and Serverless: A match made in the Cloud

Gillian Armstrong
@virtualgill





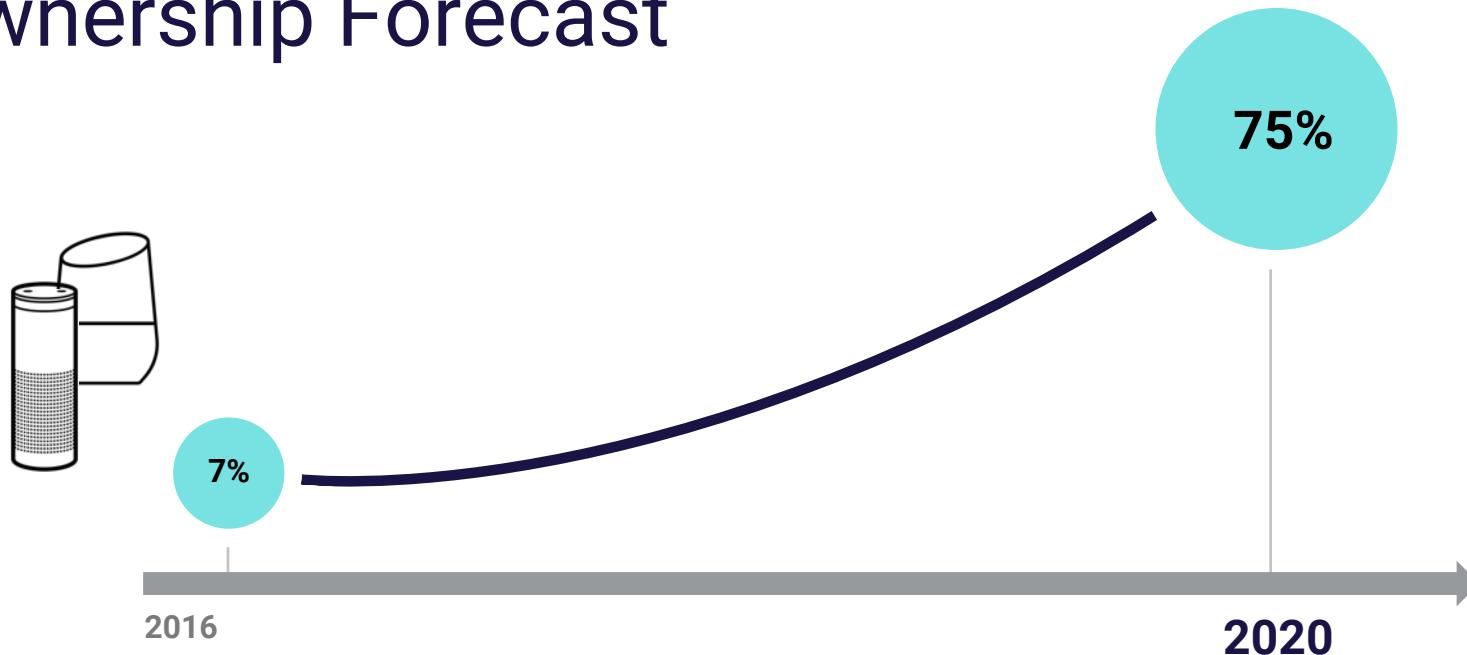
Gillian Armstrong

@virtualgill

Technologist
Cognitive Technologies

Liberty IT

Smart Speaker US Household Ownership Forecast



SOURCE: US Census, Gartner,
Edison Research, Voicebot.ai

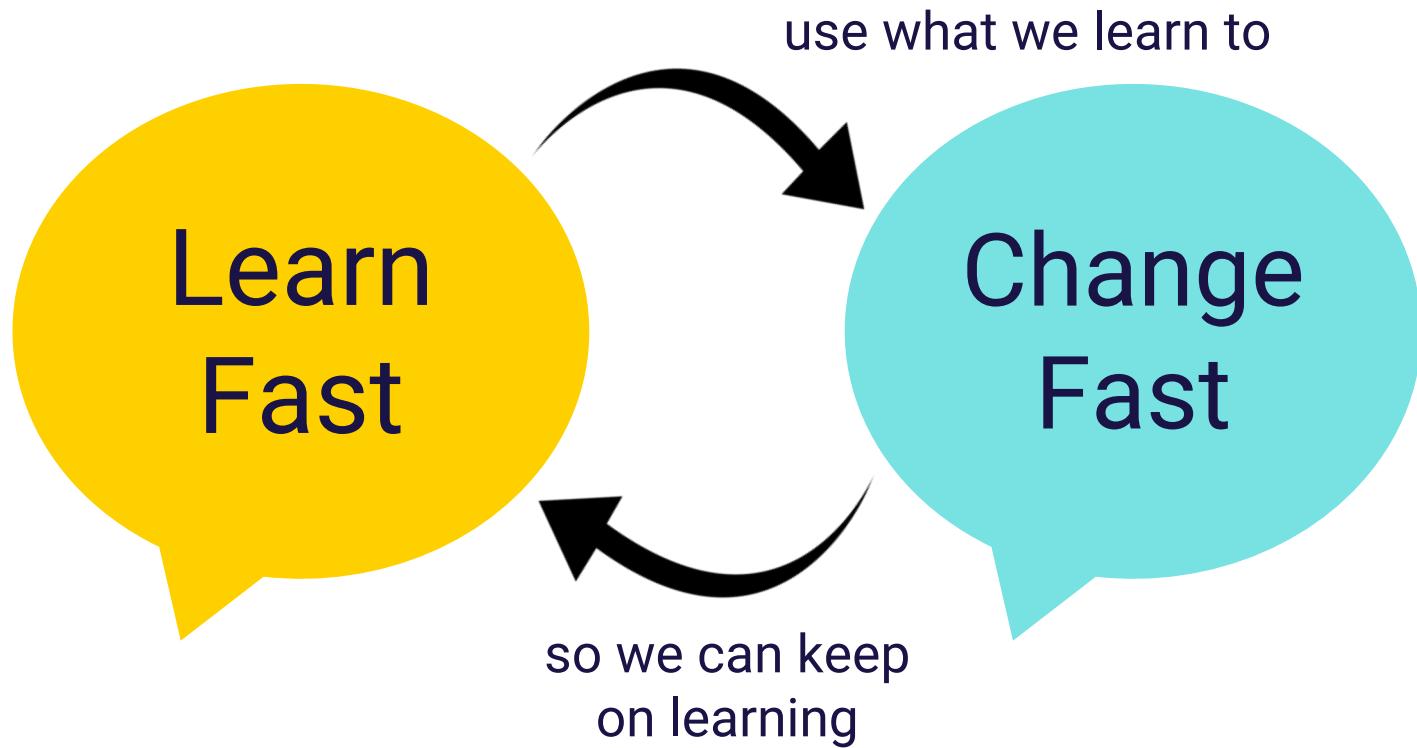
Smart Speaker US Household Ownership Forecast



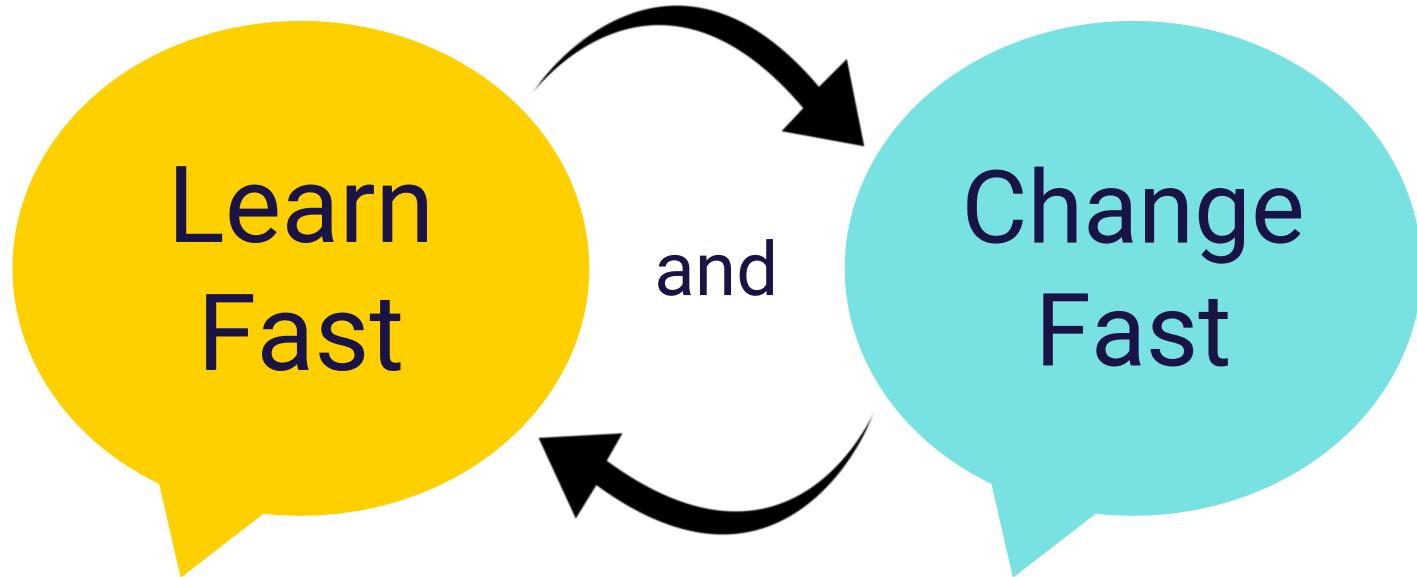
SOURCE: US Census, Gartner,
Edison Research, Voicebot.ai

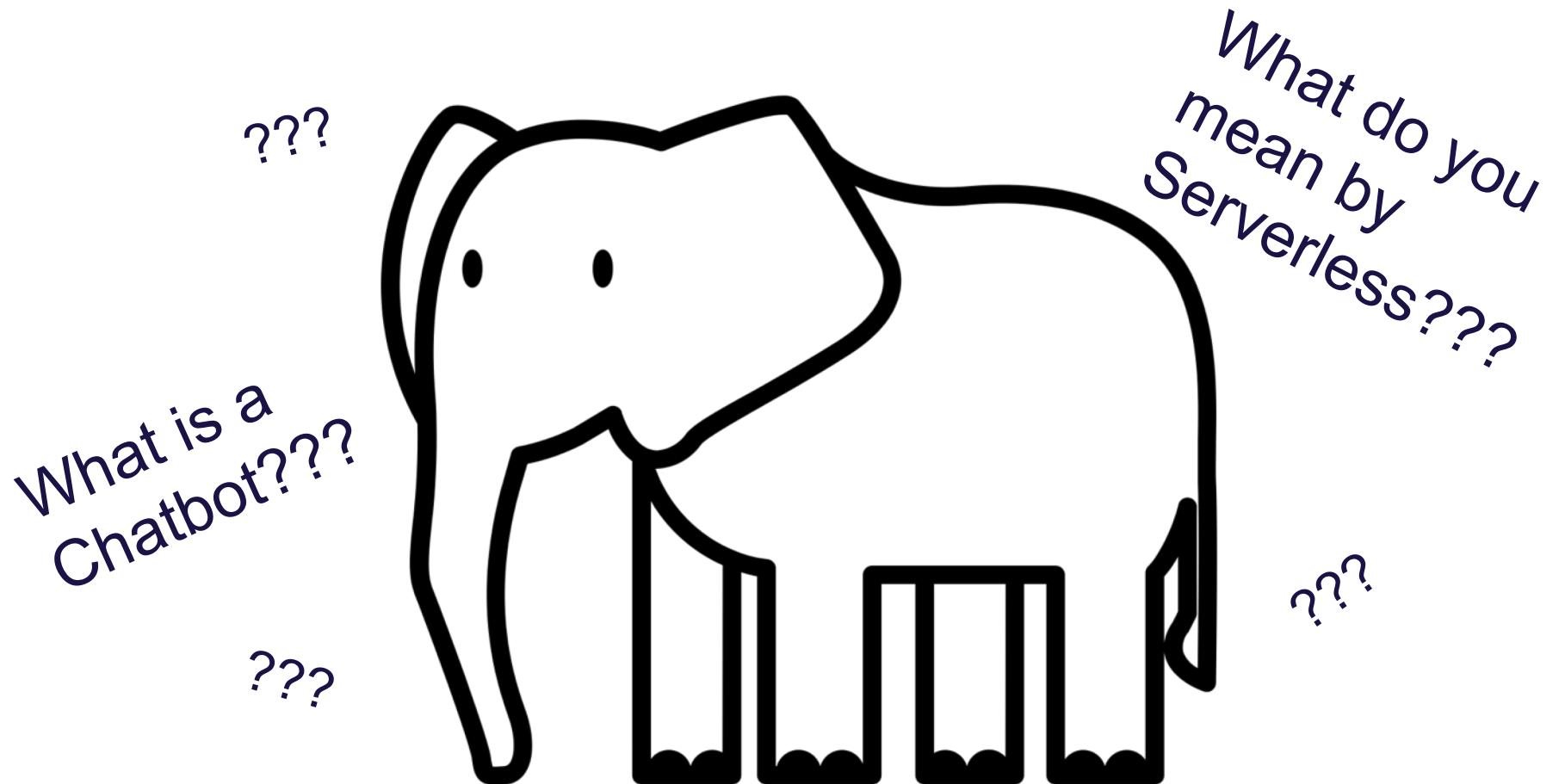
The world is
changing
fast

So we need to
be able to
learn fast

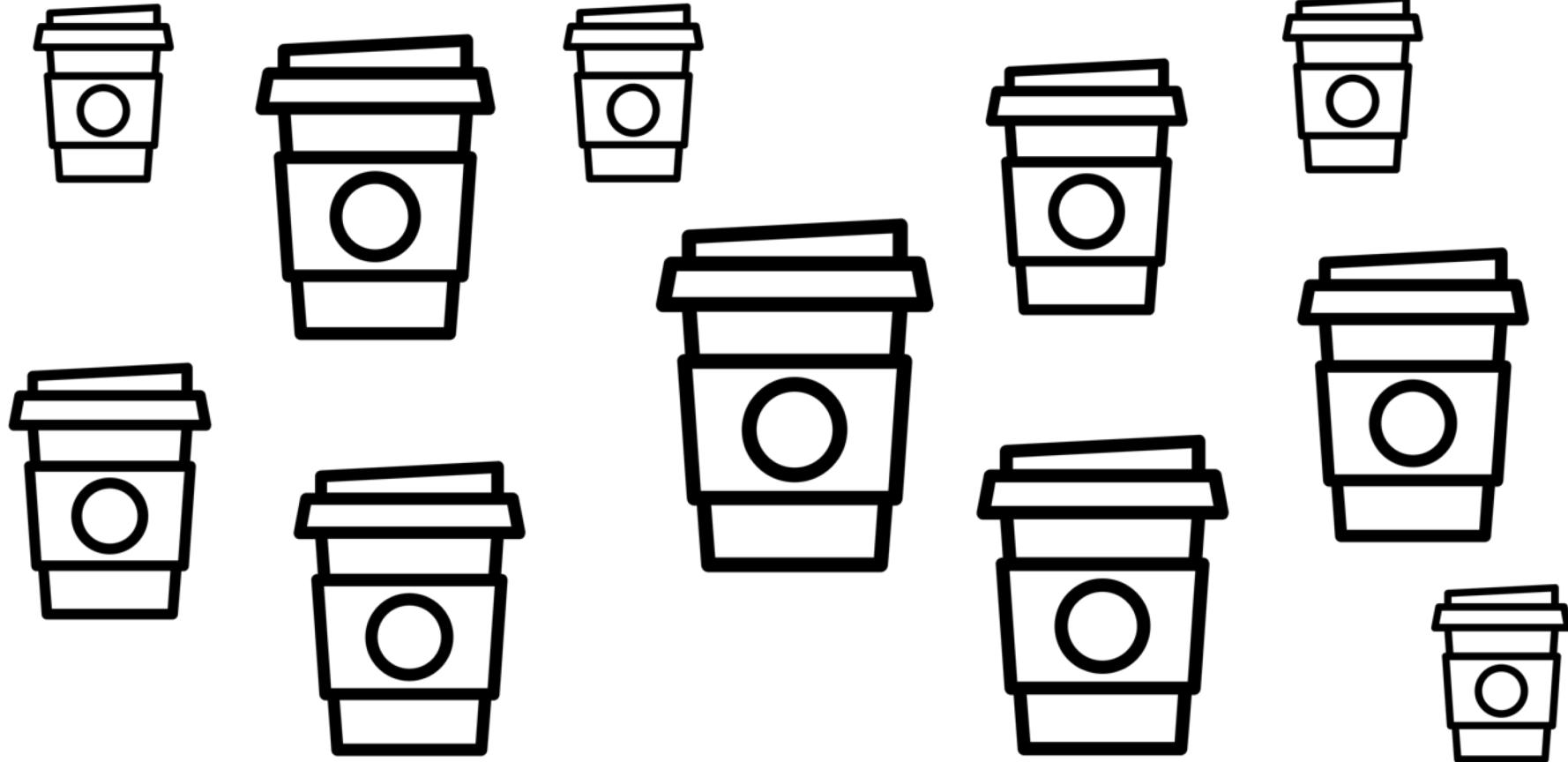


Both **Chatbots** and **Serverless** let us









For the purposes of this presentation...

**A Serverless solution is one that costs
you nothing to run if nobody is using it
(excluding data storage cost)**

- Paul Johnston

For the purposes of this presentation...

A Chatbot is something you can interact with conversationally using natural language

- me

Digital Assistant BETA

Notifications 1

To Do

Approvals 1

About Me

Assistant BETA

Feedback

Hi there. What can I help you with today?

Do you know who Sofia Martinez is?

Sofia Martinez
Cognitive Technologies Team
Software Developer
Portsmouth, NH

what else can you do?

Here are a few examples of things you can say:

- What is the help desk number?
- My account is locked
- My payslip is wrong
- What is the link for ESS?
- How do I refer a friend?
- How can I apply for maternity leave?

I'm always learning to do new things, so be sure to check back regularly.

If you have an idea for something you'd like me to be able to do, let me know by saying 'I have an idea'.

Type your question here...

About Liberty Employee Center SBU or Department myConnections

View All News

Incentive Information

Incentive award information via the Compensation Statement.

12 Likes 5 Comments 12 Shares

Designing a Digital Workplace

Advancements to Liberty Mutual Insurance's digital technology are making it easier for employees to get the information they need to do their job.

12 Likes 5 Comments 12 Shares

Mobile Defender • Apr 4

Get Information. Whenever.

myConnections

What's going on?

My Places View all Activity

Katherine Longnameton commented on Designers + Developers = Better Together (and Happy Customers) Gain insights on methods and... in User Experience a minute ago

Katherine Longnameton commented on Designers + Developers = Better Together (and Happy Customers) Gain insights on methods and... in User Experience a minute ago

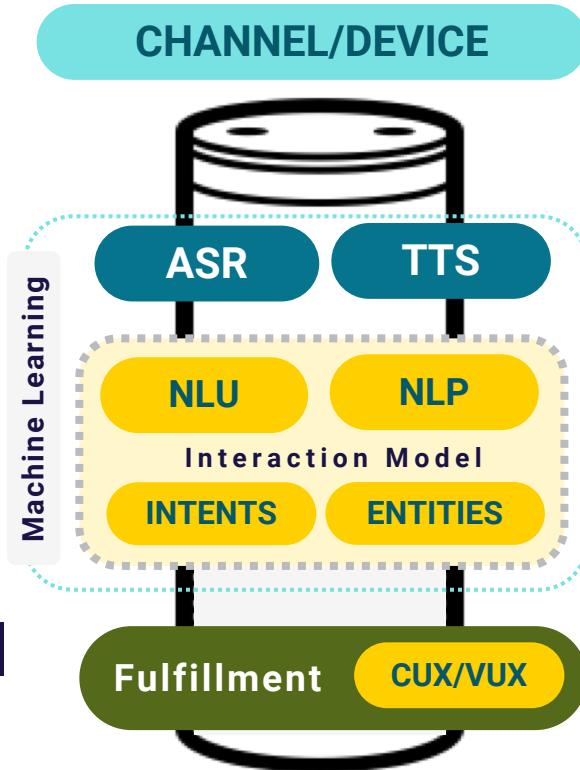
Katherine Longnameton commented on Designers + Developers = Better Together (and Happy Customers) Gain insights on methods and... in User Experience a minute ago

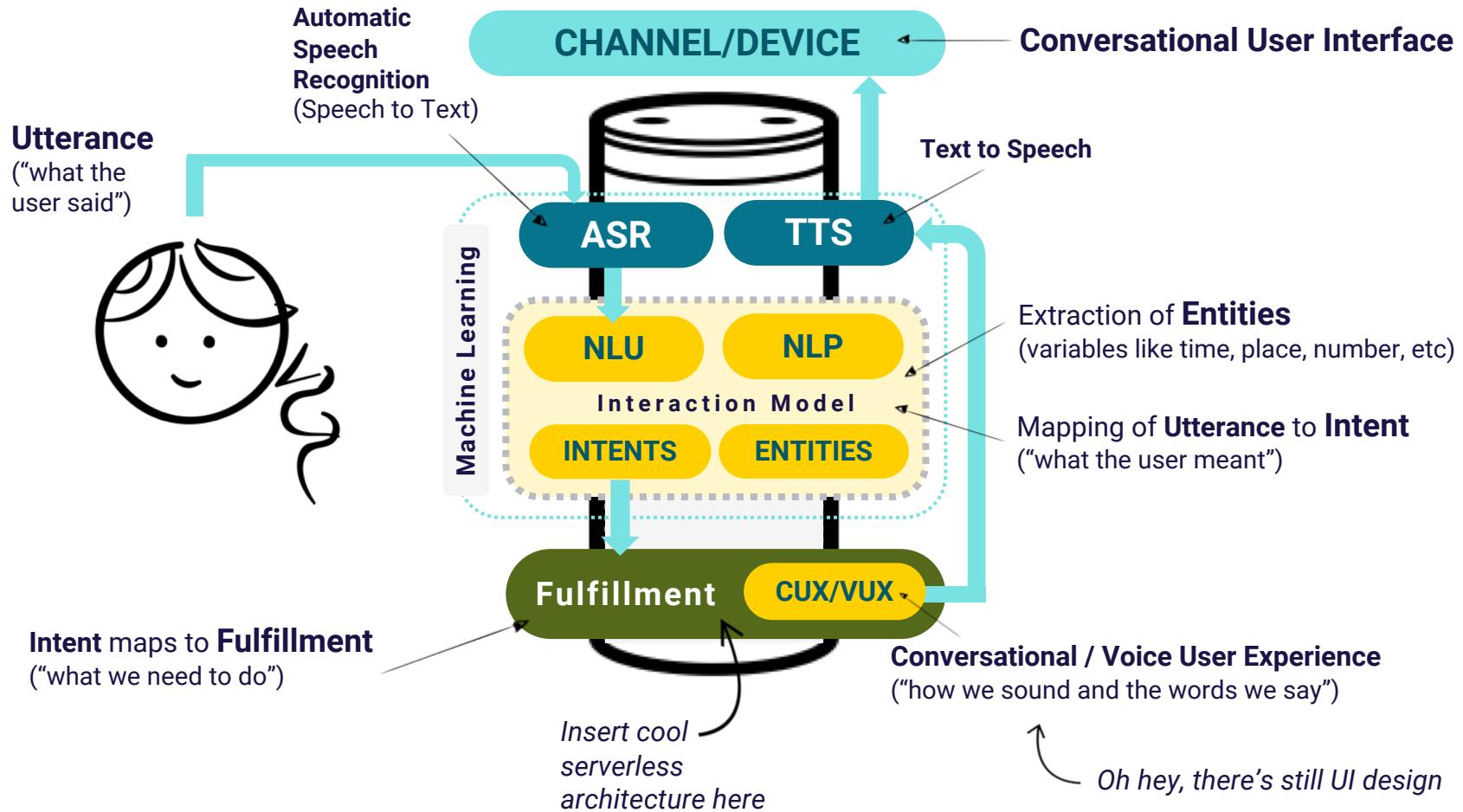
Katherine Longnameton commented on Designers + Developers = Better Together (and Happy Customers) Gain insights on methods and... in User Experience a minute ago

Katherine Longnameton commented on Designers + Developers = Better Together (and Happy Customers) Gain insights on methods and... in User Experience a minute ago

Katherine Longnameton commented on Designers + Developers = Better Together (and Happy Customers) Gain insights on methods and... in User Experience a minute ago

A very quick overview of a Chatbot using Conversational AI

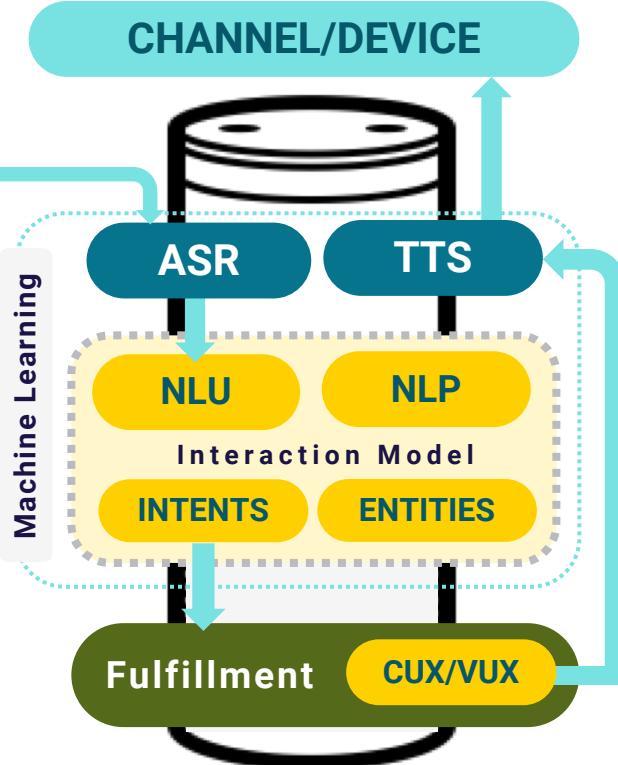




What we need to do...



1 Design it so
a Human can
understand it



2 Train your Bot
to understand
Humans

3

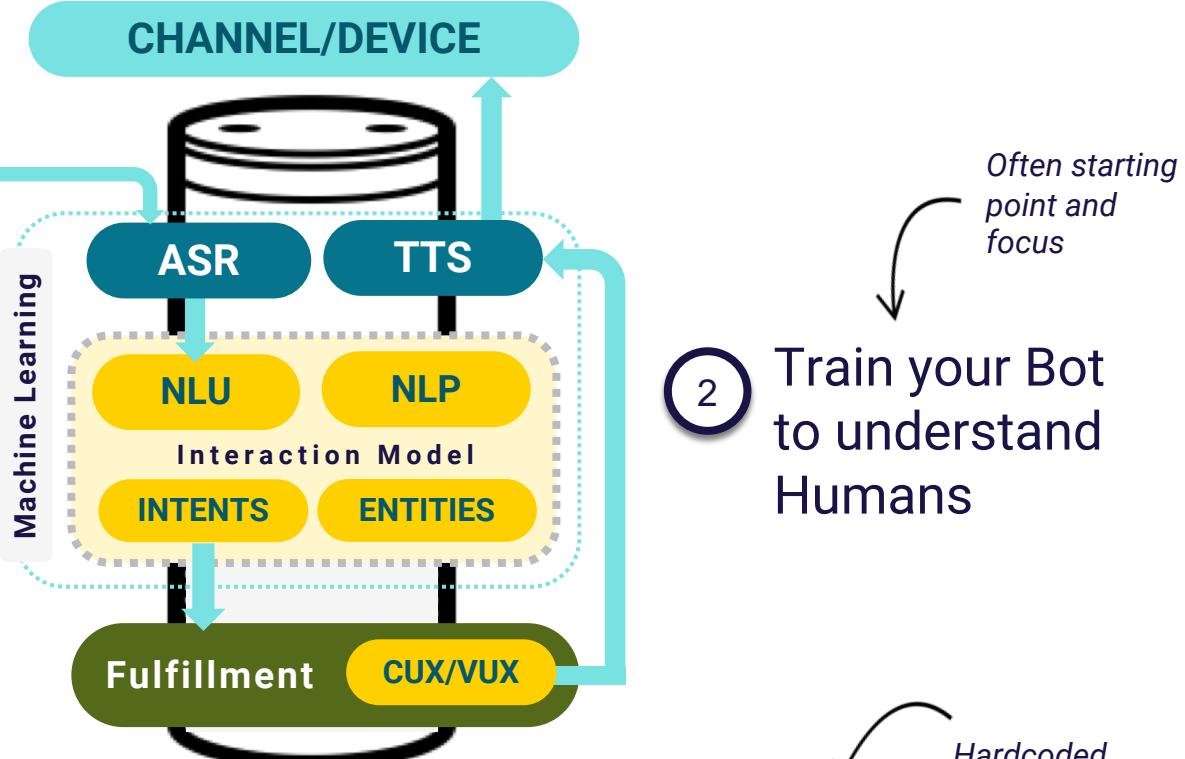
Write code to make
it do something!

What we need to do...



- 1 Design it so a Human can understand it

Forgotten



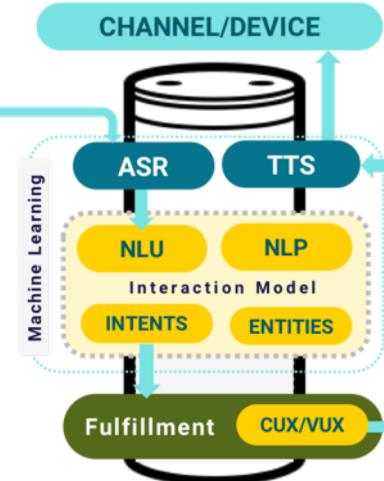
- 2 Train your Bot to understand Humans
- 3 Write code to make it do something!

Often starting point and focus

Hardcoded and then panic-written after that well-received demo

1

Design it so a
Human can
understand it



2 Train your Bot
to understand
Humans

3 Write code to make
it do something!

In a chatbot,
**Conversation is
the Interface**

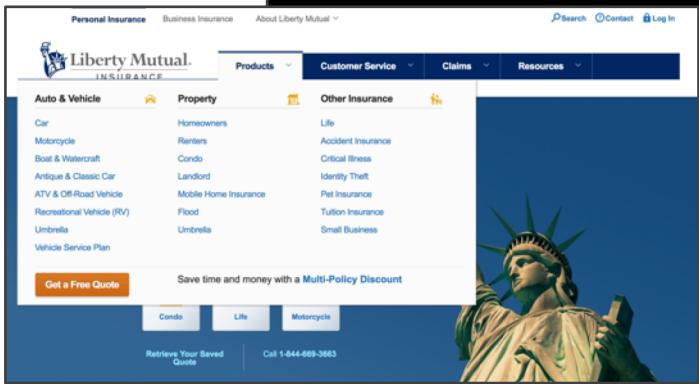
Not a CSS
and HTML
problem!



Source: <http://hosanna1.com/>

Up to now we've
kept designing
based on what
came before

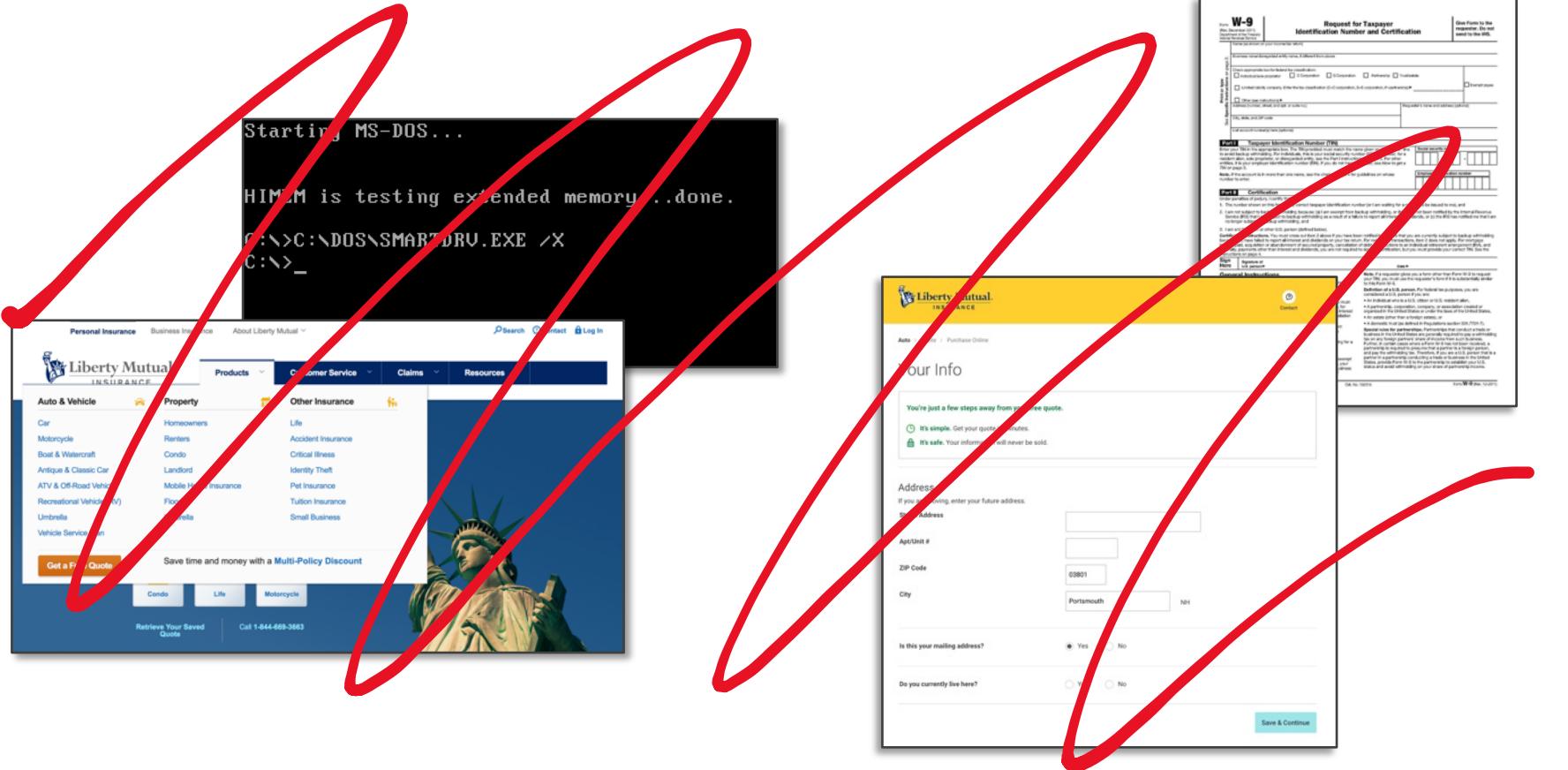
We've carried the concept of menus and commands into our graphical user interfaces

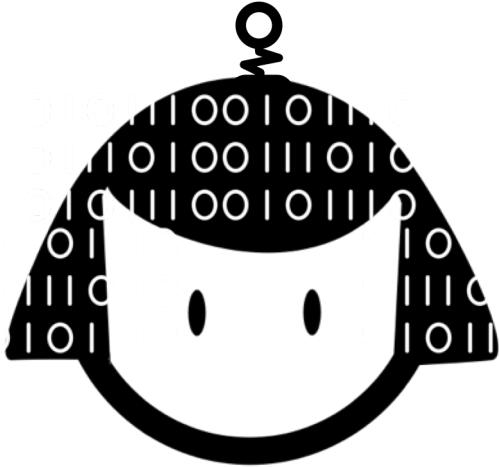


```
Starting MS-DOS...  
  
HIMEM is testing extended memory...done.  
C:\>C:\DOS\SMARTDRV.EXE /X  
C:\>_
```

We base our web forms on the paper forms that came before

The image shows two side-by-side forms. On the left is a 'Request for Taxpayer Identification Number and Certification' (W-9) form from the Internal Revenue Service. It includes fields for name, address, telephone number, and various checkboxes for business types like 'Individual proprietor', 'Corporation', 'Partnership', and 'Institute'. A note at the bottom states: 'Remember, this is an approximate form. The IRS form must match the name placed on the W-9 with the name placed on your tax return. If you do not use the exact name on the W-9, you may be subject to a late filing penalty or a fine. If you do not file a W-9, you may be subject to a late filing penalty or a fine.' On the right is a screenshot of a Liberty Mutual insurance quote form titled 'Your Info'. It has a yellow header bar with the company logo and navigation links. Below it, there's a section for entering personal information: 'Street Address', 'Apt/Unit #', 'ZIP Code', and 'City' (with examples like '03801' and 'Portsmouth, NH'). There are also questions about mailing addresses and current residence status, each with 'Yes' and 'No' radio buttons. At the bottom is a blue 'Save & Continue' button.

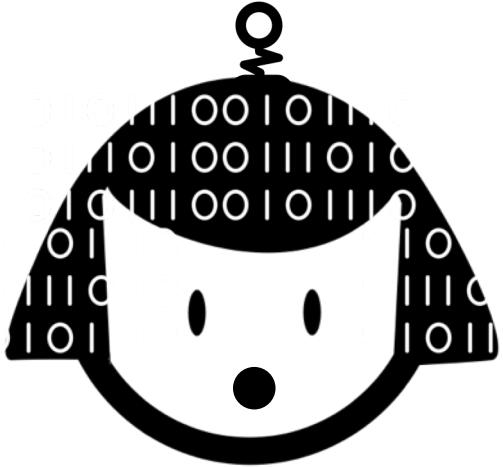




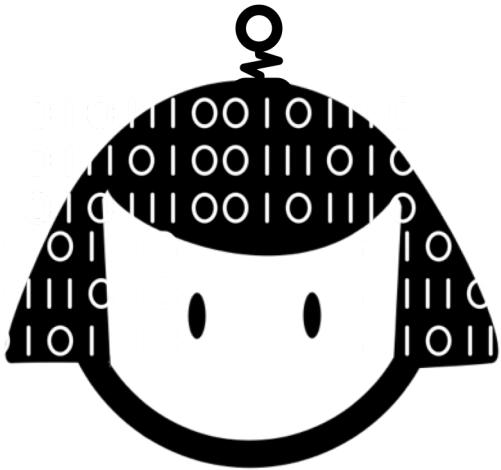
**People are the original
conversational interface**

**Model your design on a
human, not a website**



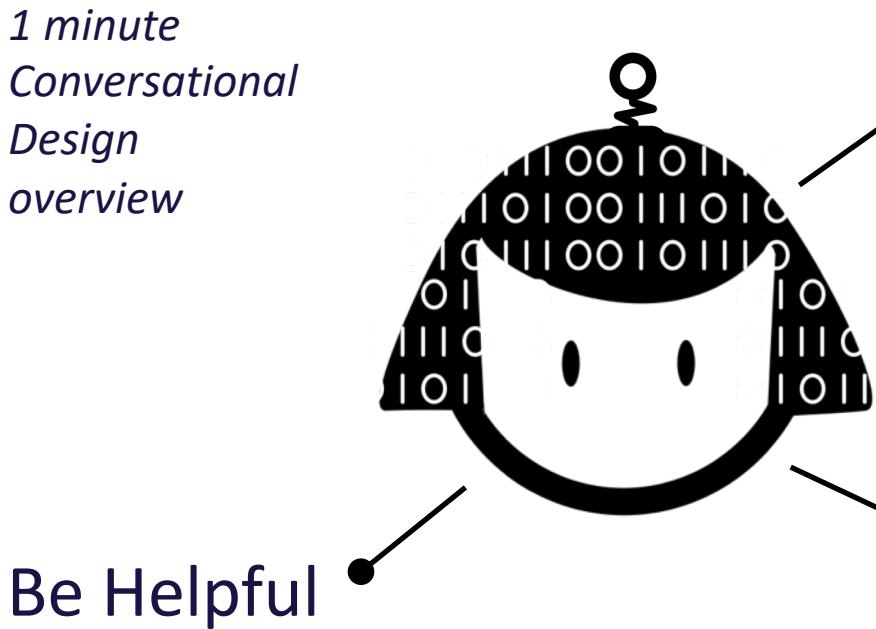


When you start with thinking about trying out a design you may discover that a chatbot isn't even the right thing to build...



But assuming it is...

*1 minute
Conversational
Design
overview*



Be Helpful

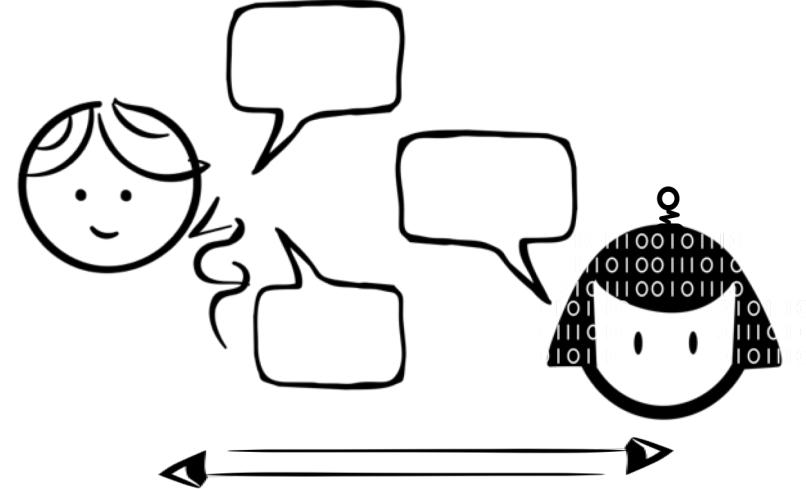
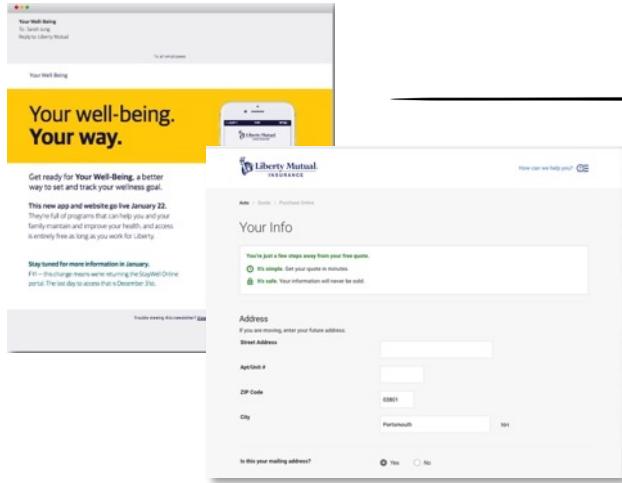
- Proactively offer help
- Get people back on track
- Repetition is annoying
- Repetition is annoying

Be Clear

- Keep your personality consistent
- Keep answers short and to the point
- Set expectations – what can you do
- Reflect back key information and confirm before critical actions.

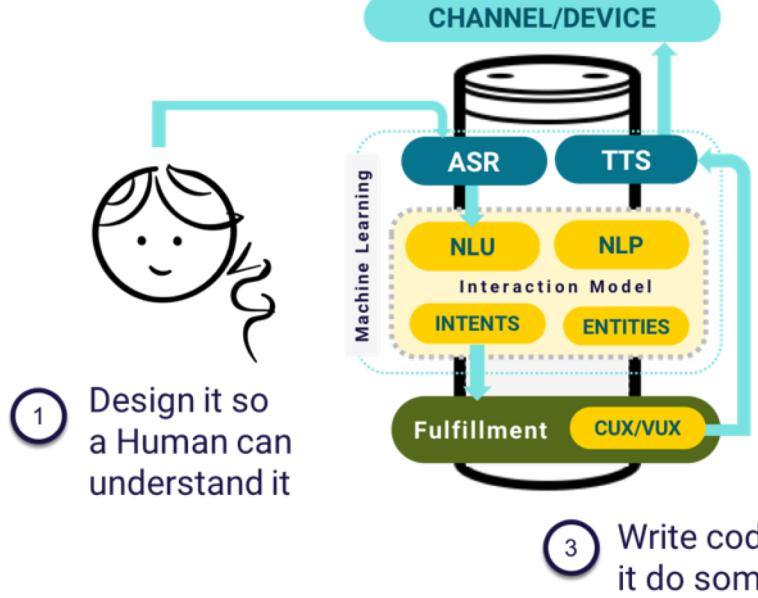
Be Nice

- Take the blame – in a conversation the human is never wrong
- Chatbots aren't smarter than a human
- Be understanding if people don't want to talk to you – offer alternatives.



The Web is about *Telling*
the user what to do

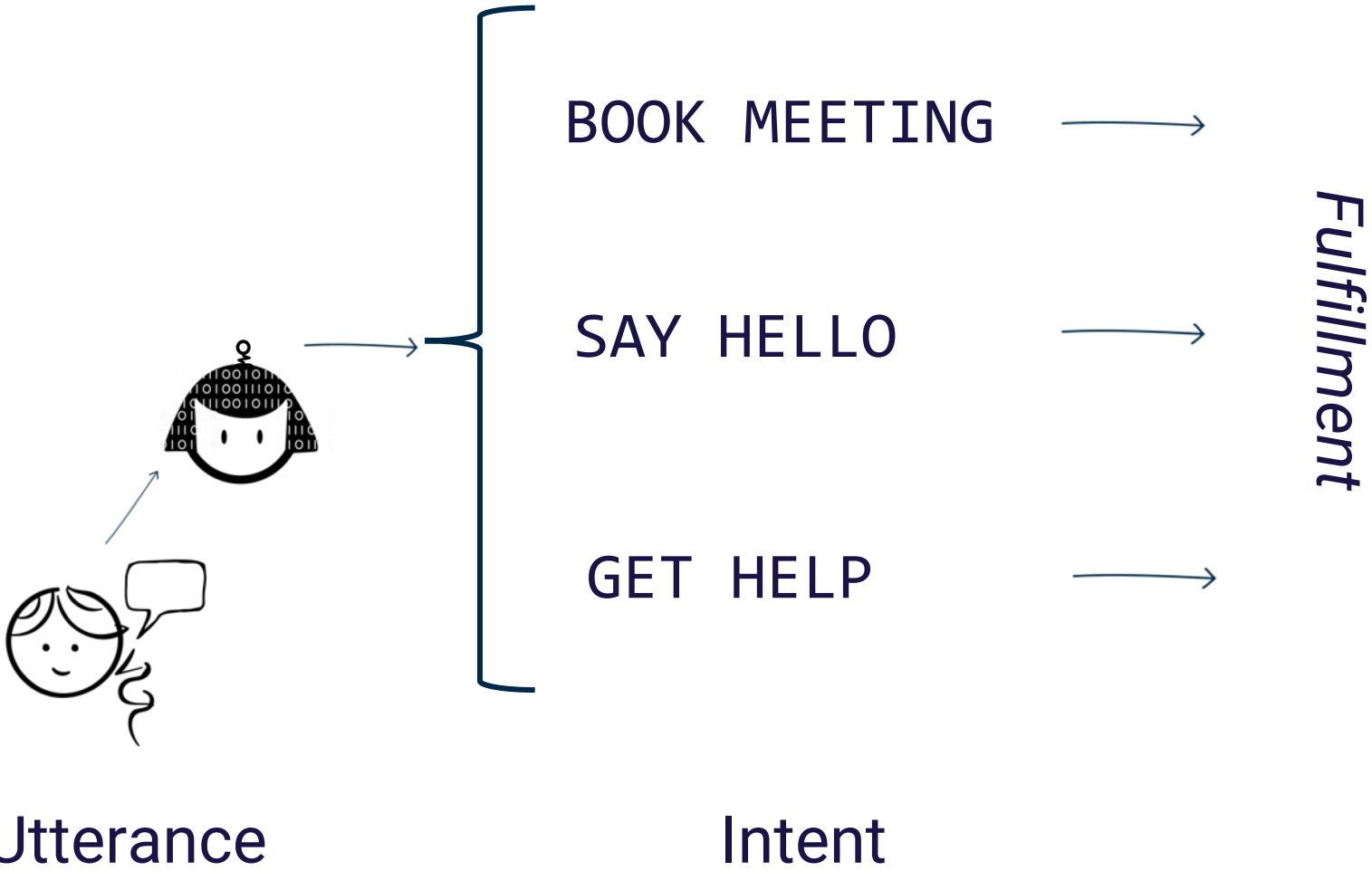
Conversational UI is
about **Listening** to what
the user wants to do



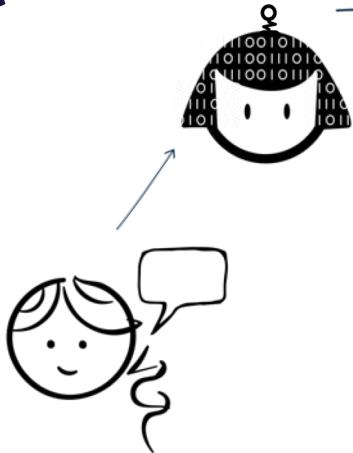
2 Train your Bot to understand Humans

Amazon Lex Console

What the Web Tutorials told me building a chatbot looked like...



As simple
providing a set
of sample
utterances!



Utterance

Intent

BOOK MEETING

Book Meeting at <TIME> on <DATE>
Set up Appointment
Create event on <DATE>
Set up Meeting
Book me a meeting at <TIME>
...

SAY HELLO

Hello
Hi there
Hey
Good Morning
...

GET HELP

Help
What can you do
Help me
I need help
...

Amazon Lex Console

SET OF INTENTS
Represents the functionality the bot has

The screenshot shows the AWS Lambda interface for managing a bot named "DemoAssistantBot". On the left, there's a sidebar with tabs for "Editor", "Settings", "Channels", and "Monitoring". The "Editor" tab is selected. Under "Intents", there's a list of intents: "demo_BookMeetingIntent", "demo_CalendarAvailabilityIntent", "demo_GenericHelpIntent", "demo_GoodbyeIntent", "demo_GreetingIntent", and "demo_OrderDrink". Below that, under "Slot types", it says "No slots created". Under "Error Handling", there's a link to "Edit". The main content area is titled "demo_BookMeetingIntent" and shows a list of "Sample utterances". These include:

- please book a meeting at [time]
- i want to book a meeting
- i need a meeting booked
- create a calendar event
- can you create a calendar event
- can you book me a meeting with [person] at [time] on [date]
- can you book me a meeting
- book time in my calendar with [person]
- book me a meeting [time]

SAMPLE UTTERANCES
Examples of what a user might say to trigger this Intent.
This is **training data - not a regex match**.

Amazon Lex Console

SLOTS (ENTITIES)
are defined showing
type (place, name,
time, custom, etc.)

The screenshot shows the Amazon Lex console interface. At the top, there are three sample utterances:

- book a meeting with {person} at {time}
- book a meeting with {person} {date}
- book a meeting with {person}

Below the utterances is a section titled "Lambda initialization and validation" with a checkbox for "Initialization and validation code hook".

On the right side, there is a large brace grouping the sample utterances and the Lambda validation section.

At the bottom, there is a table titled "Slots" with the following data:

Priority	Required	Name	Slot type	Version
		e.g. Location	e.g. AMAZON.US_CITY	
1.	✓	numberOfAttendees	AMAZON.NUMBER	Built-in
2.	✓	person	AMAZON.Person	Built-in
3.	✓	date	AMAZON.DATE	Built-in
4.	✓	time	AMAZON.TIME	Built-in

If the user does not supply this data
the bot knows to prompt for it.

The sample
utterances show
where to expect the
data to be supplied in
an utterance

Amazon Lex Console

FULFILLMENT location defined.
Intent and **Slot** matches will be sent to this lambda or returned directly to client.

Slots 

Priority	Required	Name	Slot type
		e.g. Location	e.g. AMAZON.US_CITY
1.	✓	numberOfAttendees	AMAZON.NUMBER
2.	✓	person	AMAZON.Person
3.	✓	date	AMAZON.DATE
4.	✓	time	AMAZON.TIME

Confirmation prompt 

Fulfillment 

AWS Lambda function Return parameters to client

Lambda function: demo-lex-bot [View in Lambda console](#)

Version or alias: Latest

Amazon Lex Console

Build to create the
Interaction Model

Publish to make
the functionality
live to users.

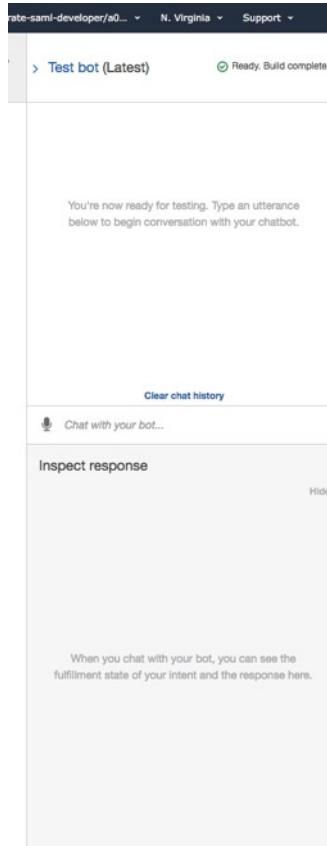
The screenshot shows the AWS Lambda interface for the Amazon Lex console. On the left, there's a sidebar with 'Intents' (demo_BookMeetingIntent, demo_CalendarAvailabilityIntent, demo_GenericHelpIntent, demo_GoodbyeIntent, demo_GreetingIntent, demo_OrderDrink) and 'Slot types' (No slots created). The main area shows a list of sample utterances for the 'demo_BookMeetingIntent':

- e.g. I would like to book a flight.
- please book a meeting at [time]
- I want to book a meeting
- I need a meeting booked
- create a calendar event
- can you create a calendar event
- can you book me a meeting with [person] at [time] on [date]
- can you book me a meeting
- book time in my calendar with [person]
- book me a meeting [time]
- book me a meeting with [person]

At the top right, there are 'Build' and 'Publish' buttons. The 'Publish' button is highlighted with a blue arrow pointing to it from the text above. To the right of the buttons, it says 'Test bot (Latest)' and 'Ready. Build complete.' Below the buttons is a text input field with placeholder text 'Type an utterance below to begin conversation with your chatbot.' At the bottom right, there are 'Clear chat history', 'Chat with your bot...', and 'Inspect response' buttons.

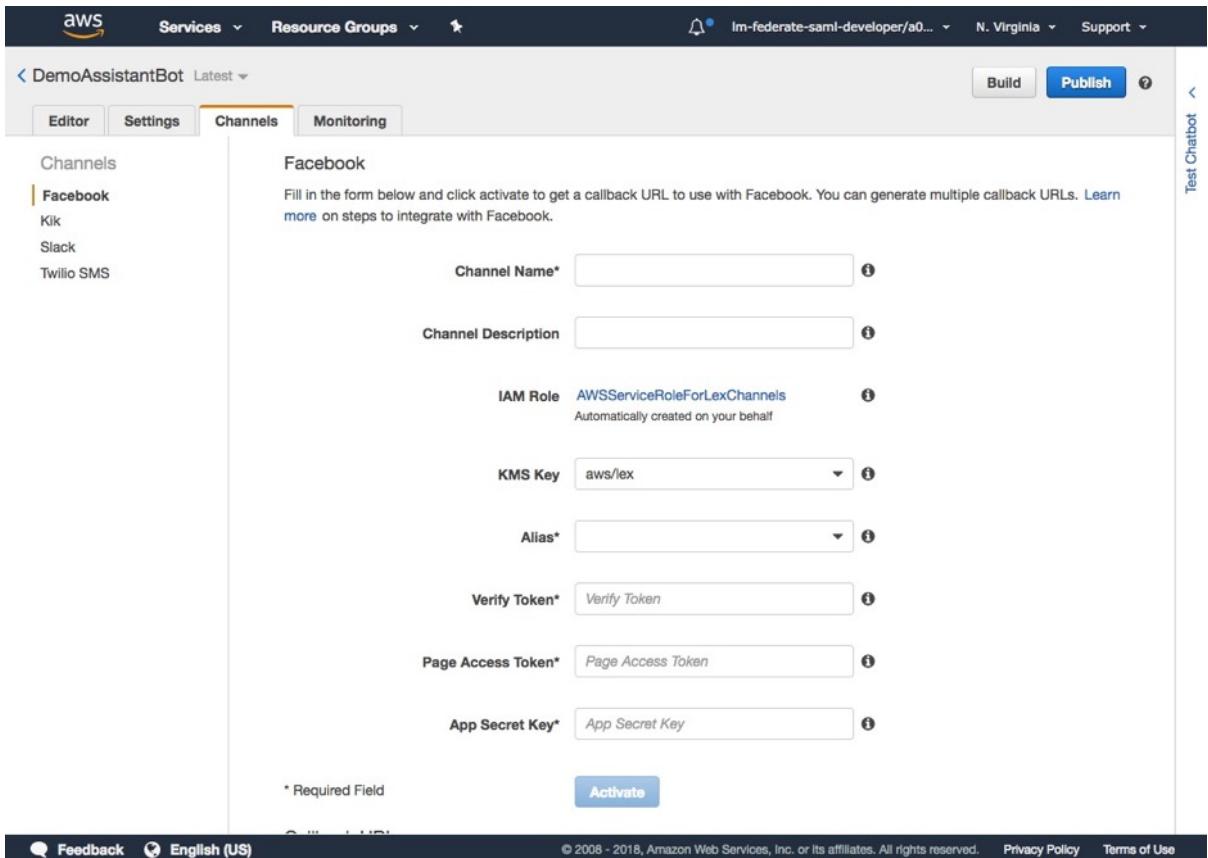
Amazon Lex Console

Built-in **Test Console** allows you to test directly here after building, but before publishing.



Amazon Lex Console

Easy to connect
your bot to popular
channels through
console.



The screenshot shows the AWS Lambda interface for a bot named "DemoAssistantBot". The "Channels" tab is selected. On the left, a sidebar lists "Facebook", "Kik", "Slack", and "Twilio SMS" as available channels. The main panel is titled "Facebook" and contains fields for configuration:

- Channel Name*: [Input field]
- Channel Description: [Input field]
- IAM Role: AWSServiceRoleForLexChannels (Automatically created on your behalf)
- KMS Key: aws/lex
- Alias*: [Input field]
- Verify Token*: Verify Token
- Page Access Token*: Page Access Token
- App Secret Key*: App Secret Key

At the bottom, there is a note: "* Required Field" and a blue "Activate" button.

Amazon Lex Console

Chatbot Done...?

DemoAssistantBot Latest

Editor Settings Channels Monitoring

Build Publish

Test bot (Latest) Ready. Build complete.

Intents

demo_BookMeetingIntent 2

Sample utterances

please book a meeting at [time]

I want to book a meeting

I need a meeting booked

create a calendar event

can you create a calendar event

can you book me a meeting with [person] at [time] on [date]

can you book me a meeting

book me a meeting

book a meeting

book a meeting with [person]

book a meeting with [person] at [time] on [date]

book me a meeting on [date] at [time]

book me a meeting on [date]

book me a meeting

book a meeting on [date]

book a meeting with [person] at [time]

book a meeting with [person] [date]

book a meeting with [person]

You're now ready for testing. Type an utterance below to begin conversation with your chatbot.

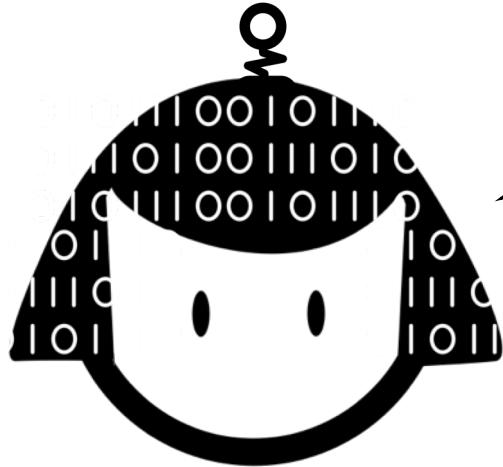
Clear chat history

Chat with your bot

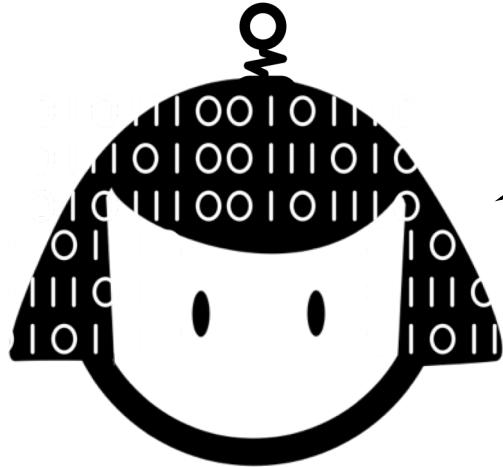
Inspect response

Hide

When you chat with your bot, you can see the fulfillment state of your intent and the response here.



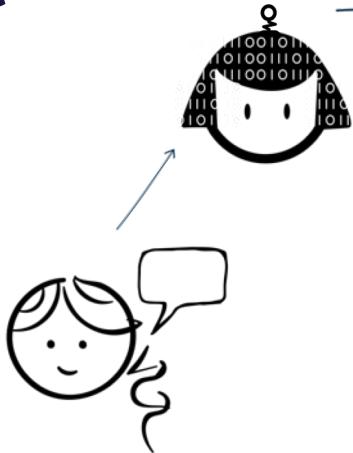
Sorry, I'm just a
baby bot. I'm still
learning.



Sorry, I'm just a
baby bot. I'm still
learning.

are you?

As simple
providing a set
of sample
utterances!



Utterance

Intent

BOOK MEETING

- Book Meeting at <TIME> on <DATE>
- Set up Appointment
- Create event on <DATE>
- Set up Meeting
- Book me a meeting at <TIME>
- ...

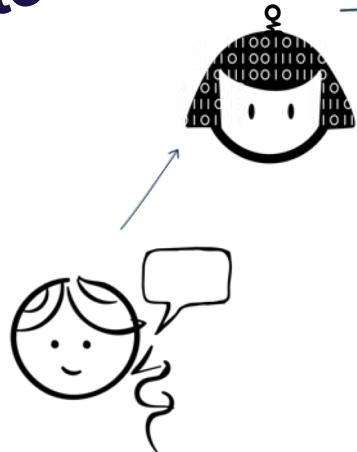
SAY HELLO

- Hello
- Hi there
- Hey
- Good Morning
- ...

GET HELP

- Help
- What can you do
- Help me
- I need help
- ...

As **complex**
as providing a
set of **sample**
utterances!



Utterance

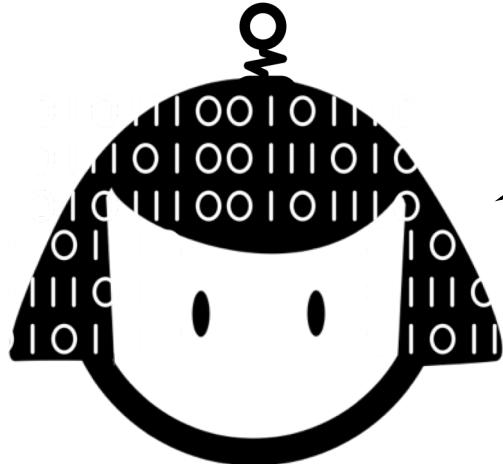
Intent

Book Meeting at <TIME> on <DATE>
Set up Appointment
Set up a meeting
Set up Meeting
Book me a meeting <TIME>
...

Hello
Hi there
Good Morning
...

Model your design on a human

Get your sample utterances from your users...
you wouldn't make up other machine learning training data!



Sorry, I'm just a
baby bot. I'm still
learning.

**Track metrics and review
conversation to improve your
Bot Interaction Model.**

**Learn
Fast**

**Change
Fast**

Reviewing Conversation Transcripts...



Look for **Vocabulary** used

Should we add new synonyms

Look for the **Utterance Structure**

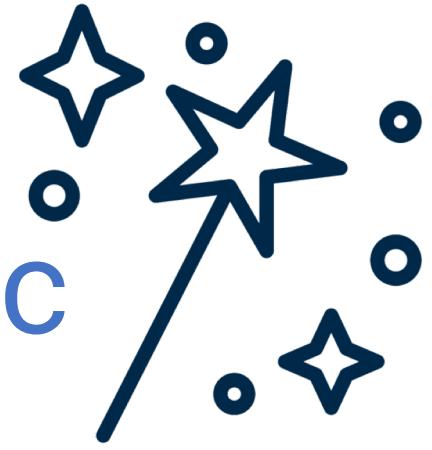
Should we add new training data

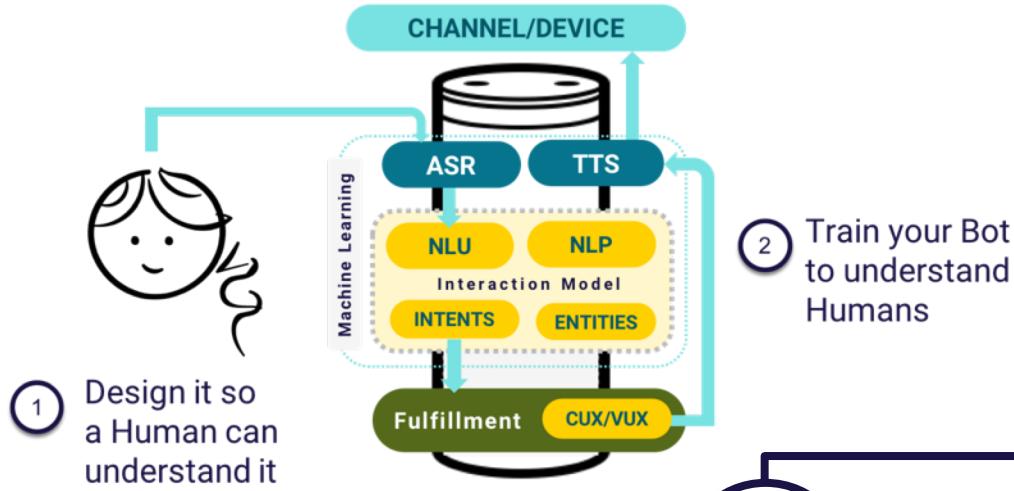
Look for **Interaction Patterns**

Can we improve the conversation flow

Change
Fast

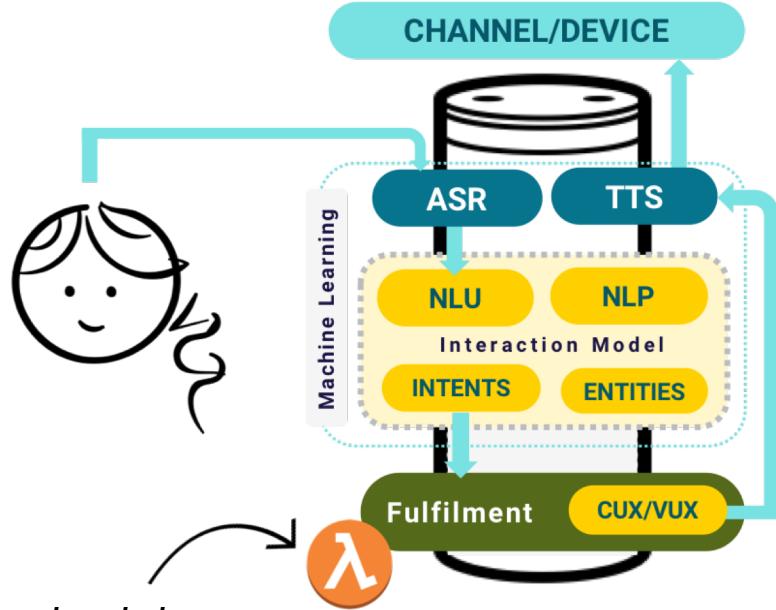
Chatbots aren't magic
(although I wish they were)



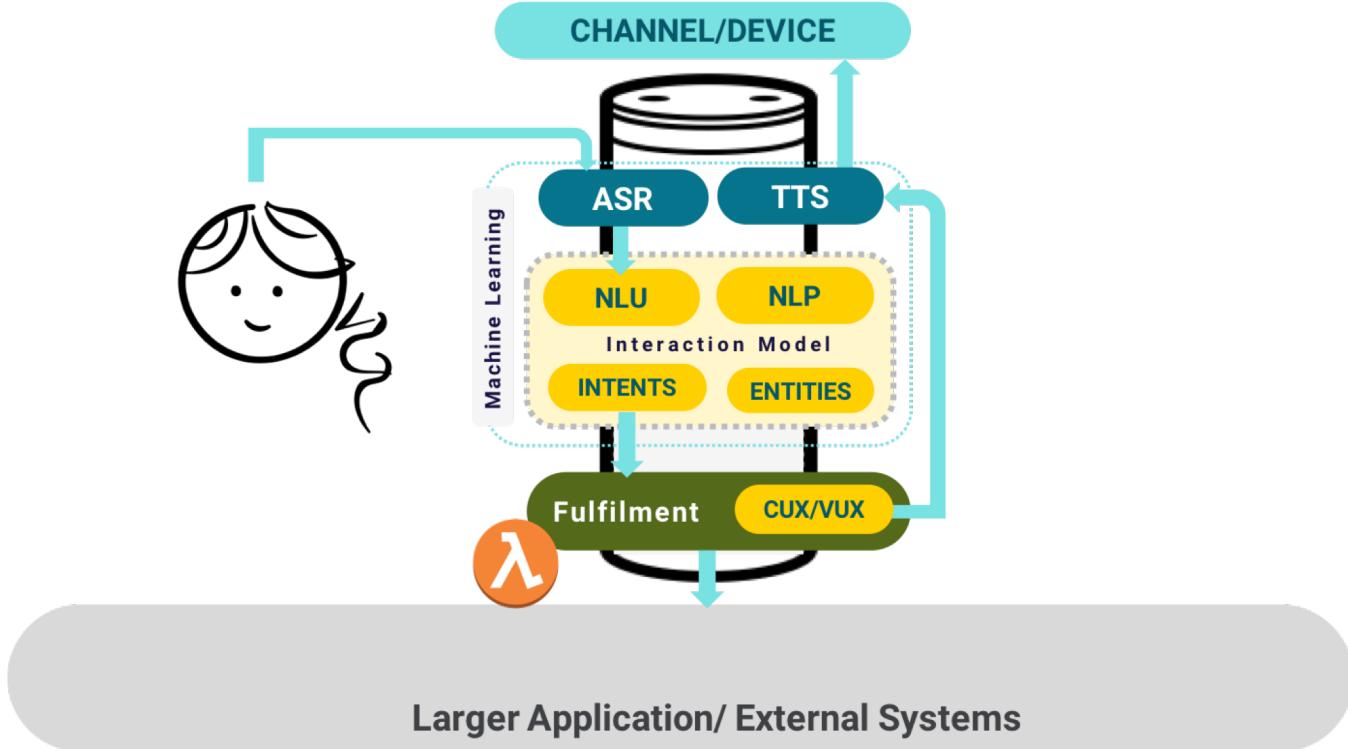


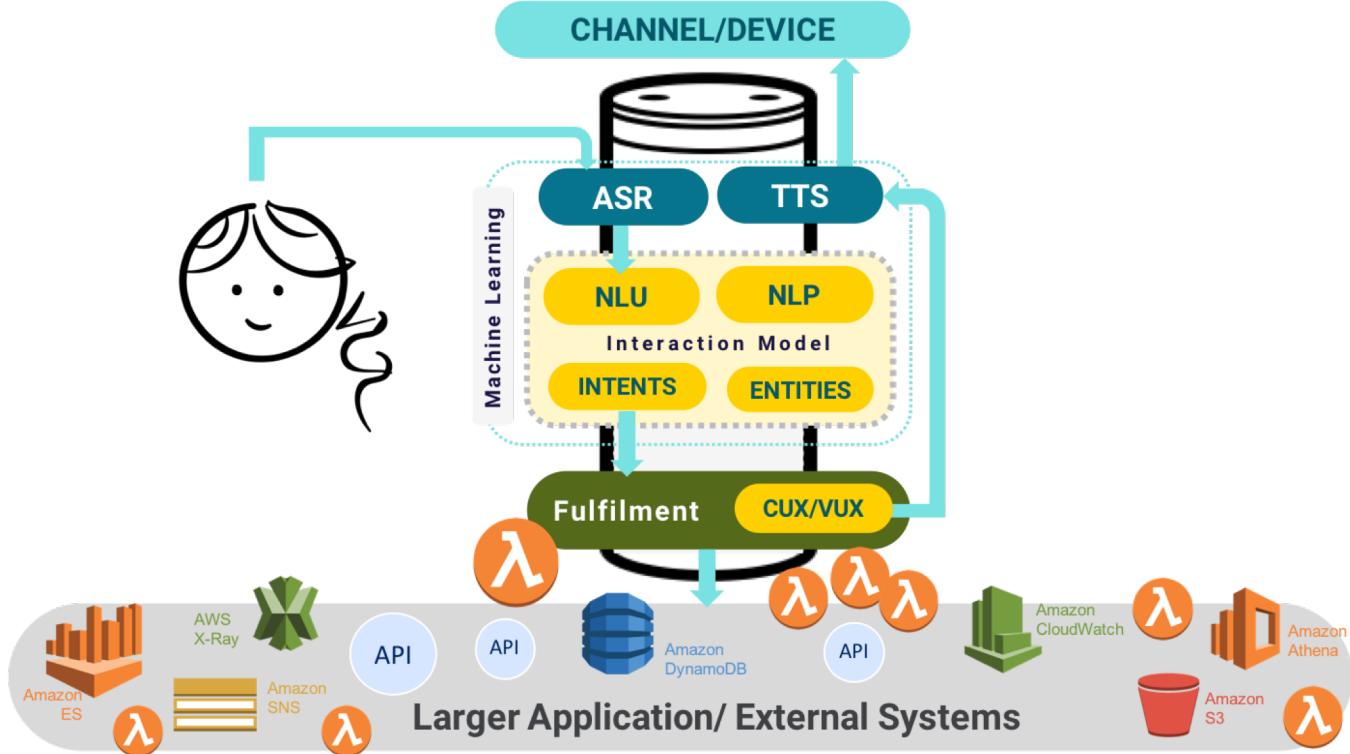
② Train your Bot to understand Humans

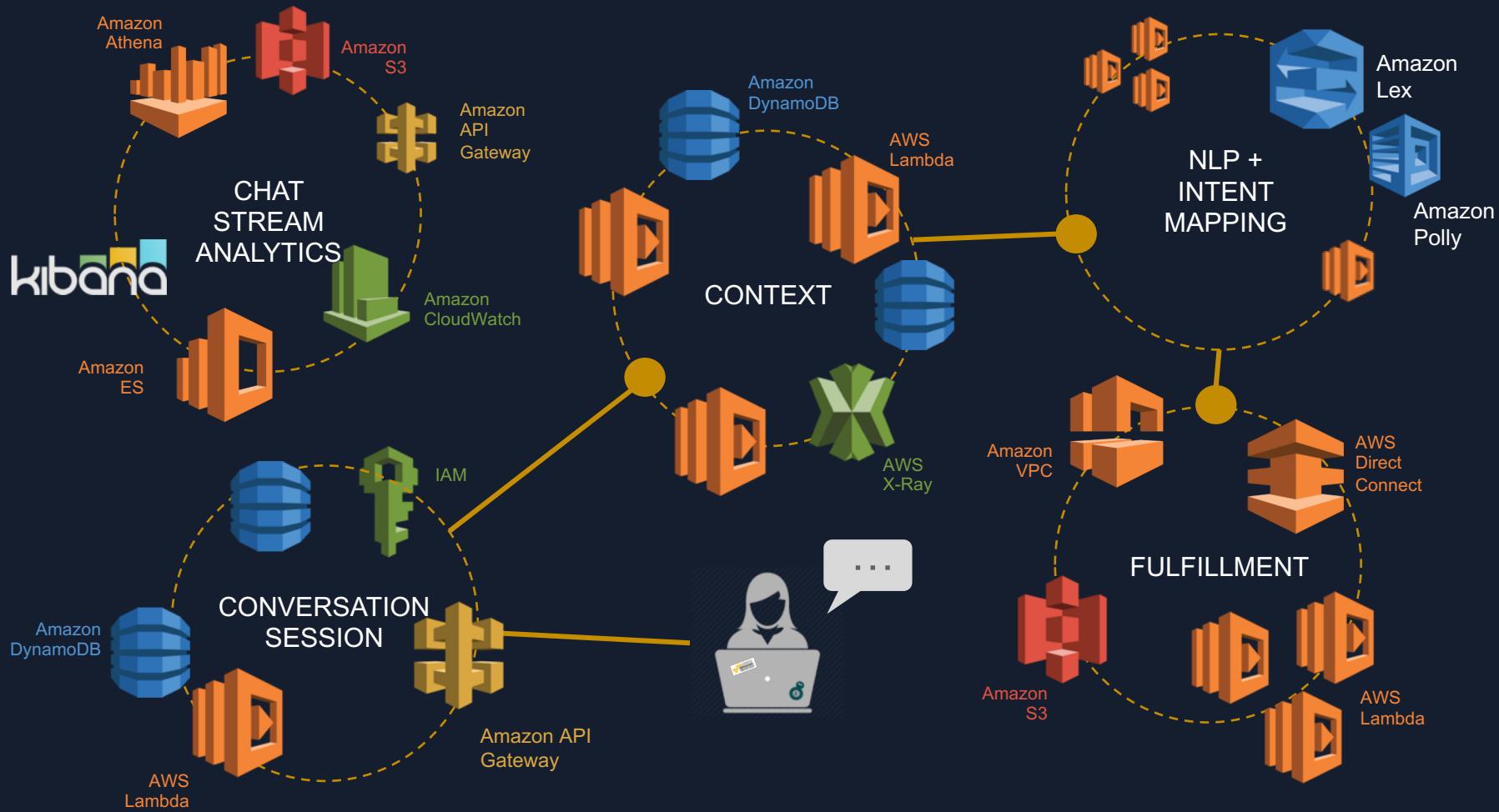
③ Write code to make it do something!



*Pop a lambda
here – and
you're done!*







THE DIGITAL ASSISTANT BOT PLATFORM ARCHITECTURE

Sessions & Utterances

Unique Users

Intent Hits Pie



229

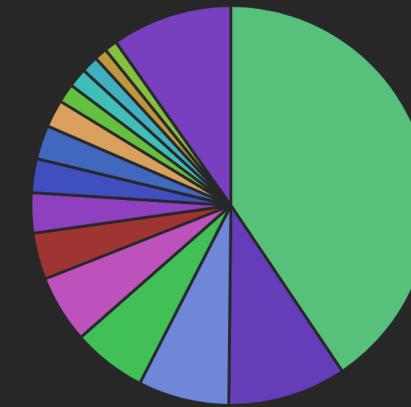
Number of Unique Sessions

166

Unique Users

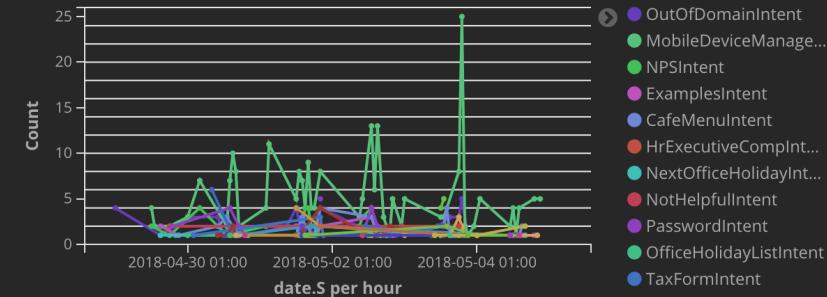
Intent Hits Table

Intent Name	Count
MobileDeviceManagementIntent	237
OutOfDomainIntent	56
CafeMenuIntent	43
NPSIntent	35
ExamplesIntent	32
QuickLinkIntent	22
PasswordIntent	19
HelpDeskIntent	16
WhoIsIntent	16
ThanksIntent	13
BrowserTroubleshootingIntent	9
NextOfficeHolidayIntent	9
HrServiceCenterGenericIntent	8
DefinitionService	6
GreetingIntent	6
	584



- MobileDeviceManagementIntent
- OutOfDomainIntent
- CafeMenuIntent
- NPSIntent
- ExamplesIntent
- QuickLinkIntent
- PasswordIntent
- HelpDeskIntent
- WhoIsIntent
- ThanksIntent
- BrowserTroubleshootingIntent
- NextOfficeHolidayIntent
- HrServiceCenterGenericIntent
- DefinitionService
- GreetingIntent
- Other

Intent hits (short timeline)

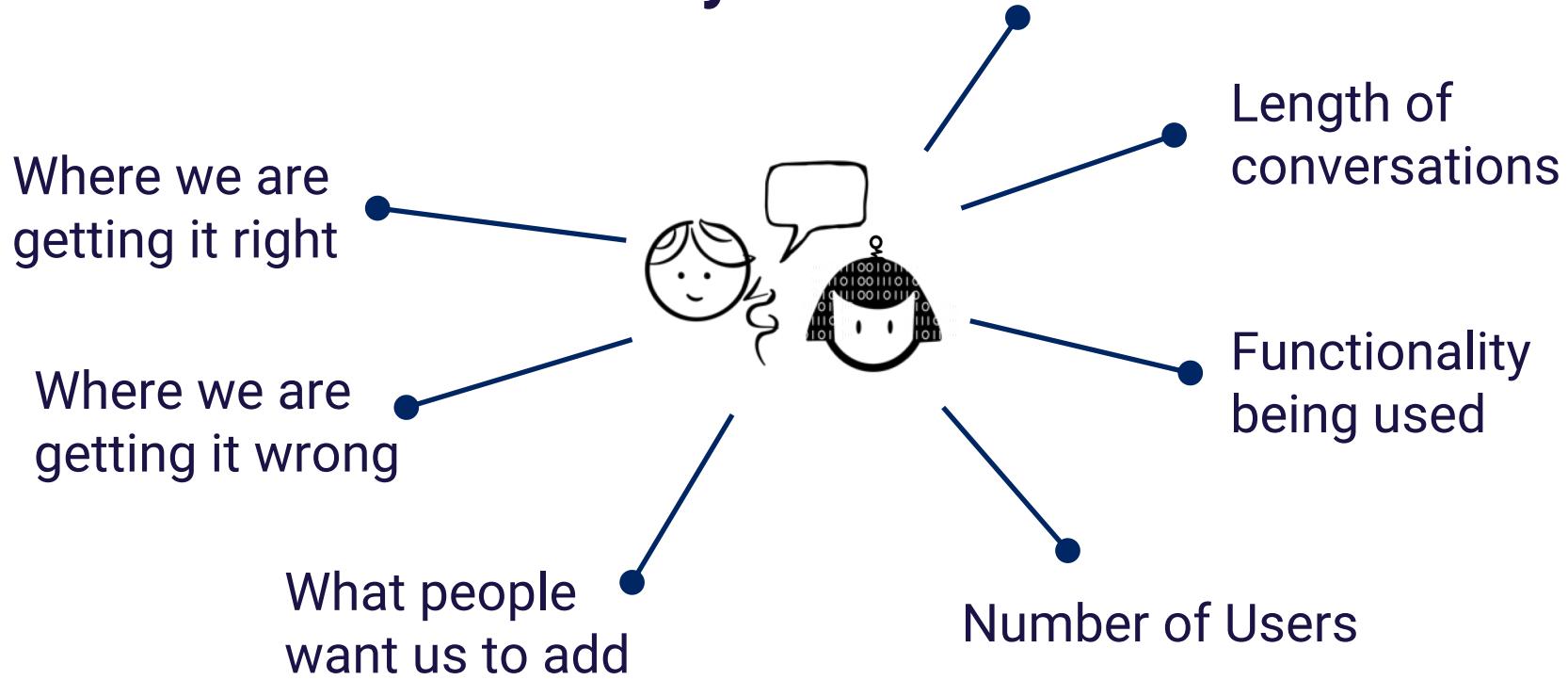


- OutOfDomainIntent
- MobileDeviceManagementIntent
- NPSIntent
- ExamplesIntent
- CafeMenuIntent
- HrExecutiveComplaintIntent
- NextOfficeHolidayIntent
- NotHelpfulIntent
- PasswordIntent
- OfficeHolidayListIntent
- TaxFormIntent
- WhoIsIntent

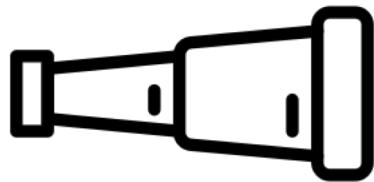
Hosting - Completed Conversations By A...

Action	Count
--------	-------

Examples of our Conversational Analytics



Observability is critical



This lets you **quickly learn** where there might be problems with your **system**.

Why Serverless??



Lower Cost
lets us

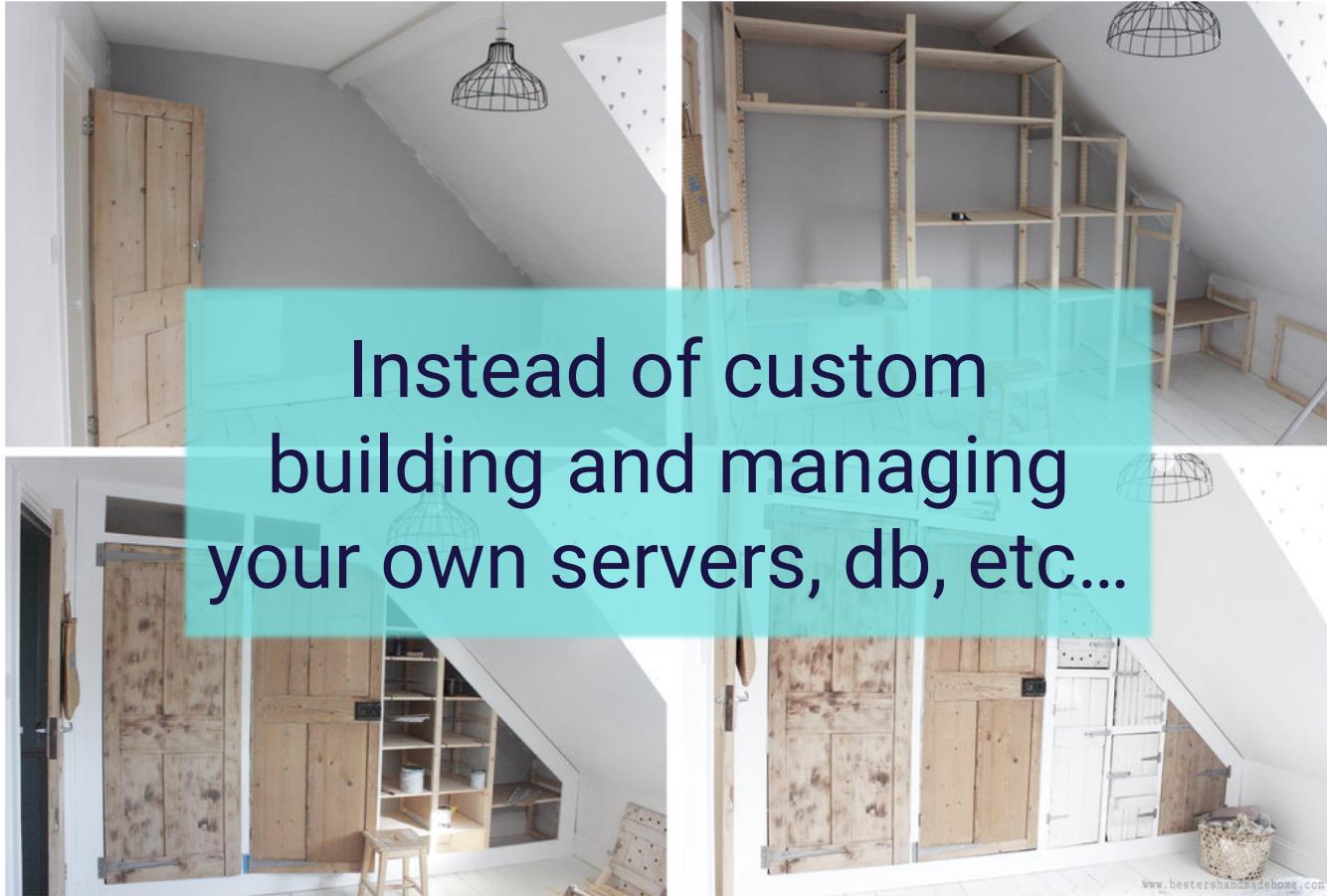


Learn
Fast

Off-loading Maintenance
lets us



Change
Fast



Instead of custom
building and managing
your own servers, db, etc...

wardrobe by www.hestershomedmadehome.com



Use standard pieces –
that someone else has
built and manages - and
put them together to get
the system you want

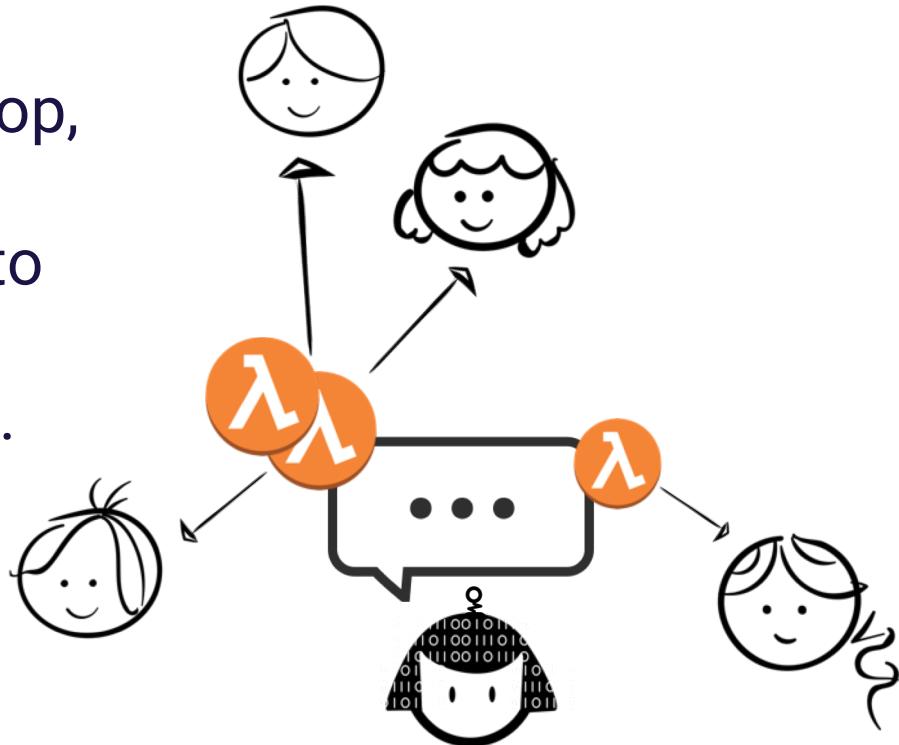


Serverless

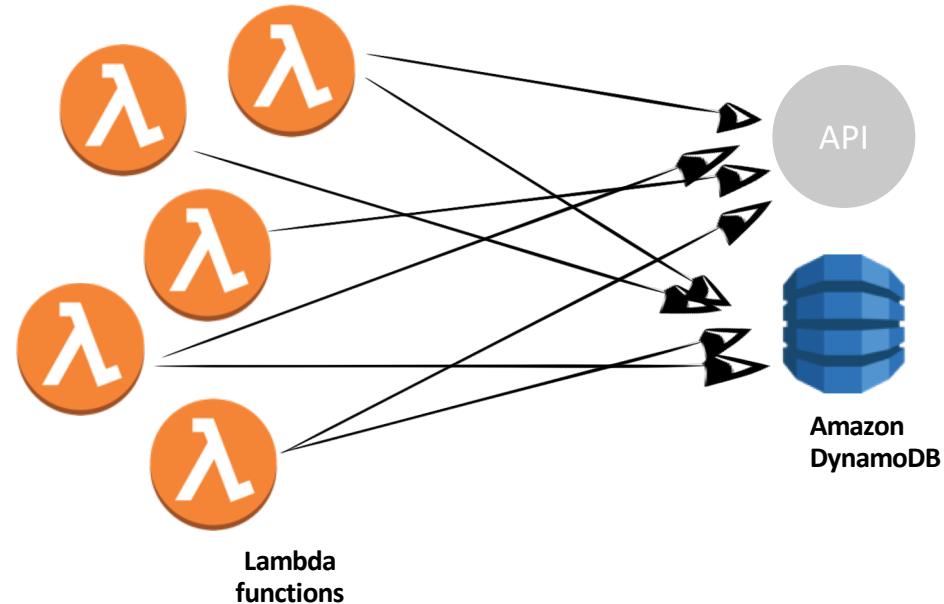


Services

Lambdas are **quick** to develop,
isolate small pieces of
functionality and allow you to
scale different **parts of the**
conversation independently.

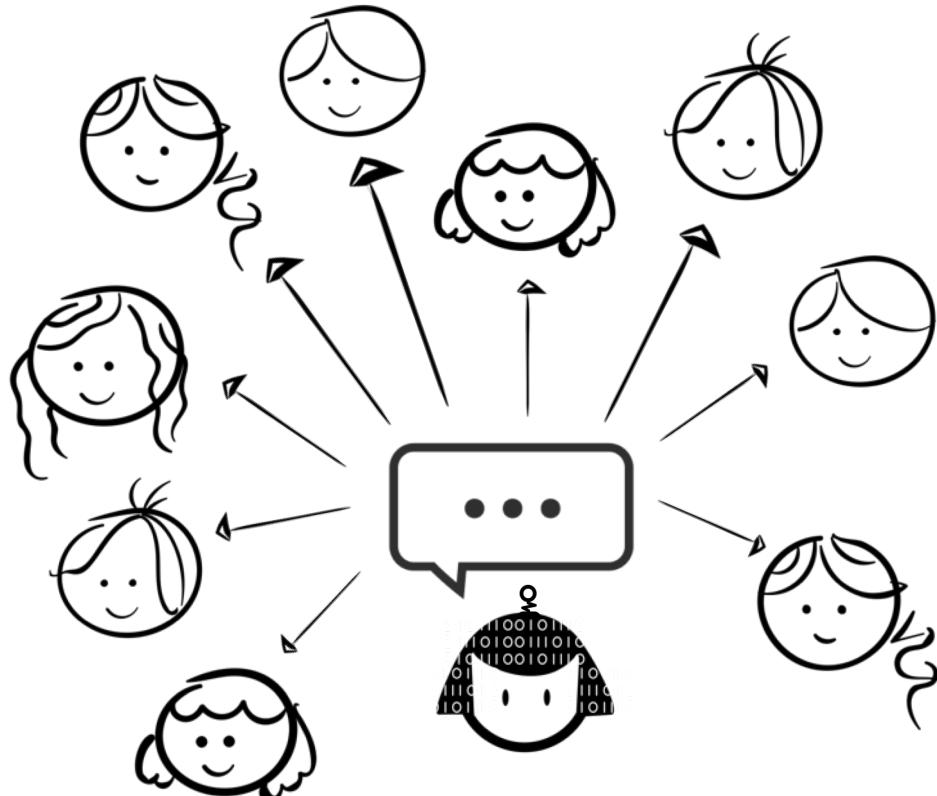
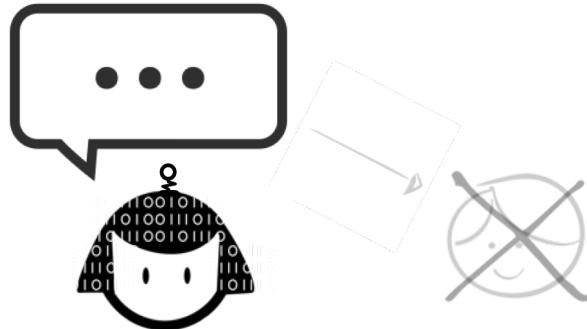


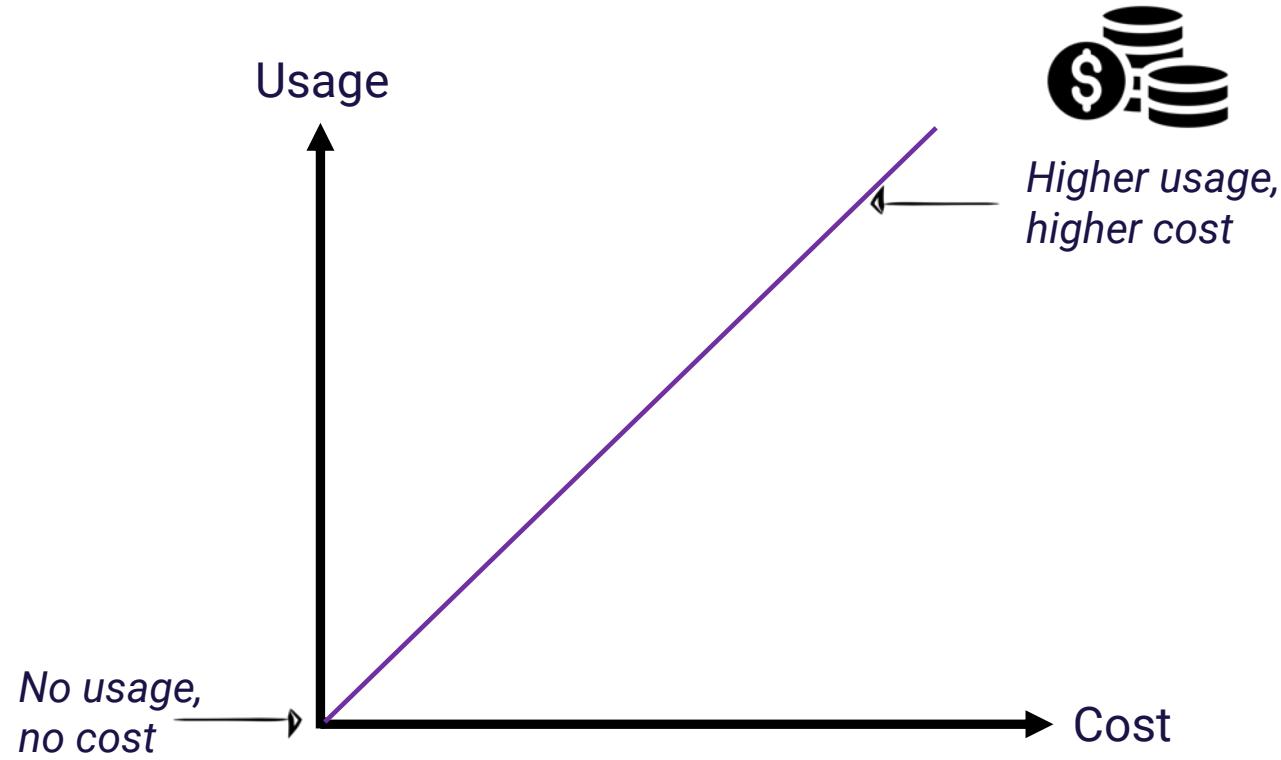
Ensure that
everything scales
- or at least know
your **limits**.

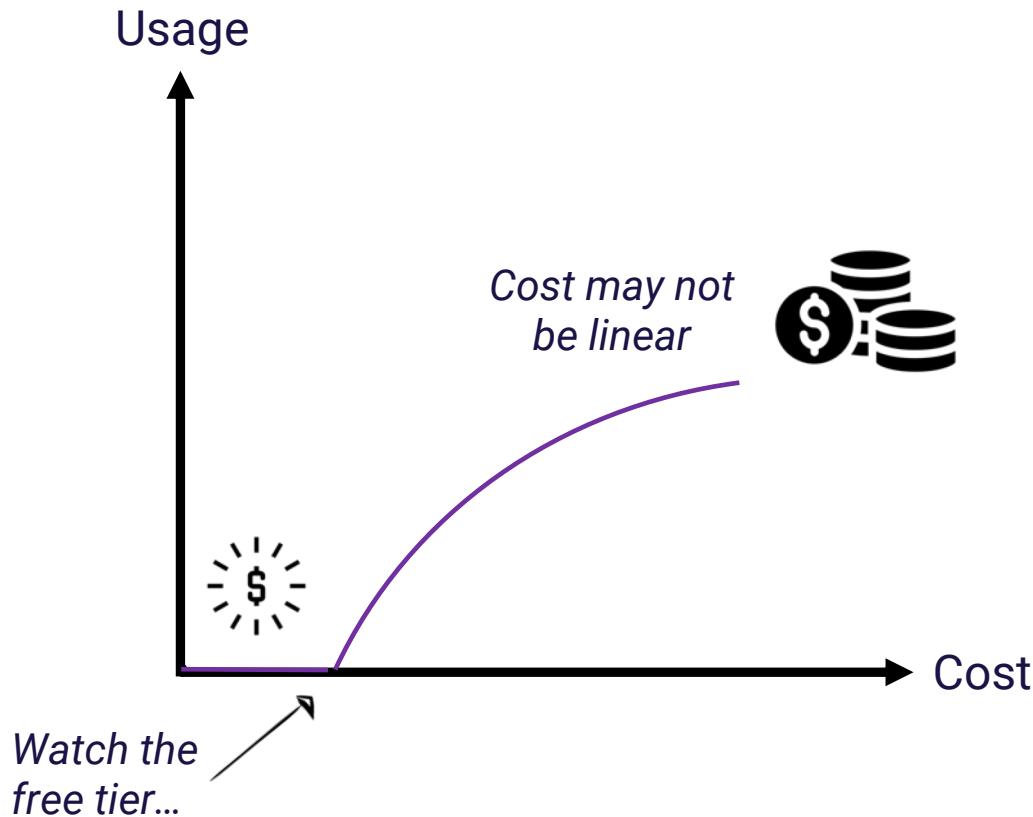


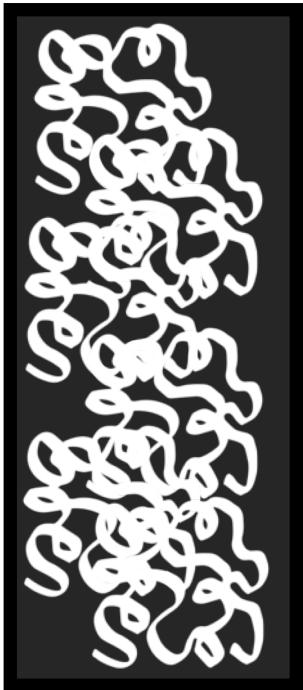
Serverless means you don't pay if **no-one** talks to your bot...

...and lets you scale seamlessly if **everyone** wants to talk to your bot!



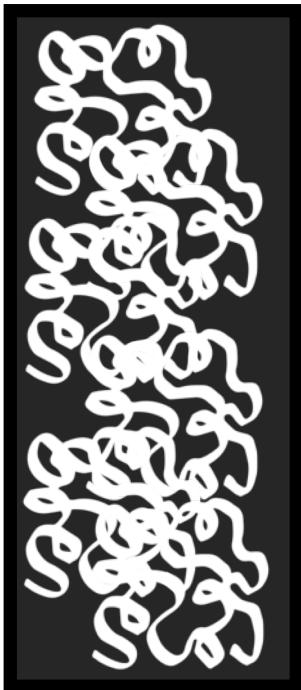






Monolith

Some notes for
those with
existing systems

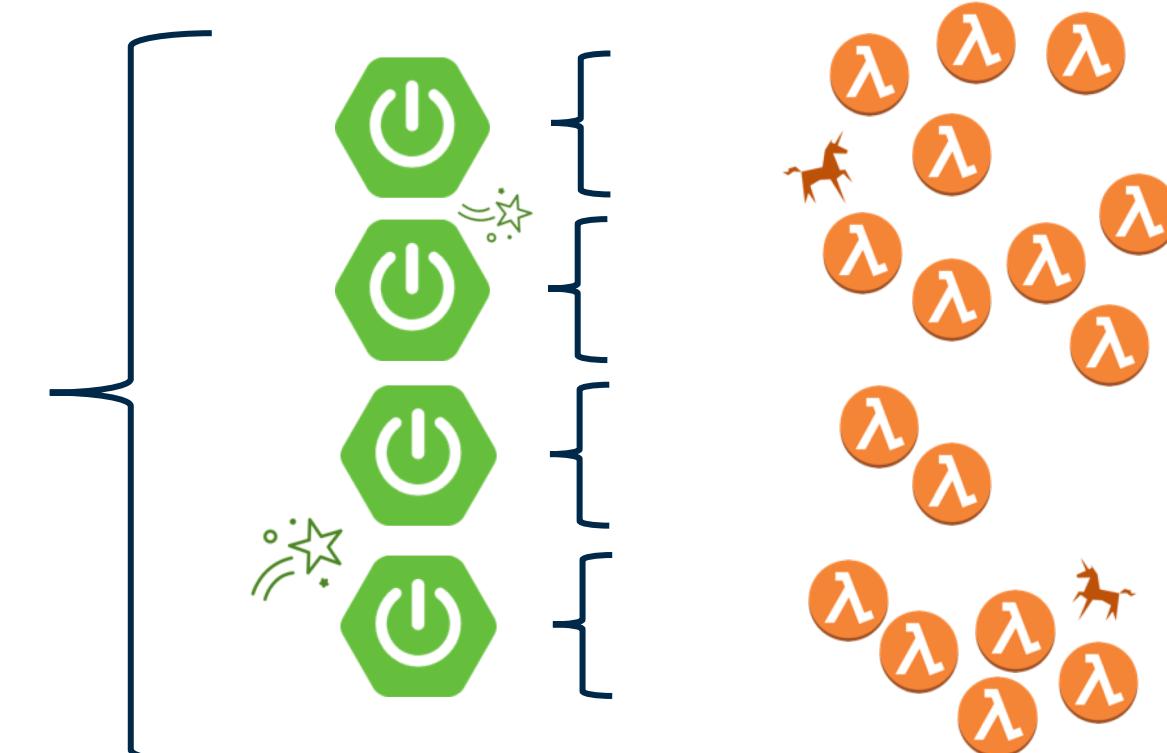
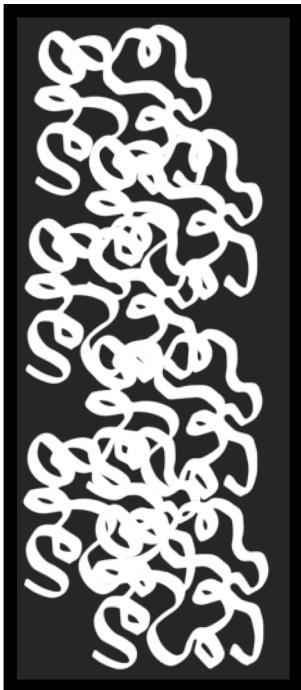


Monolith



Microservices





Monolith

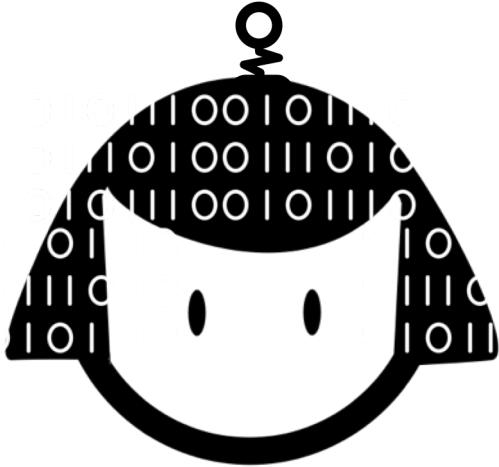


Microservices



Function as a Service

REWRITE is not a
four-letter word



**People are the original
conversational interface**

**Model your design on a
human, not a website**

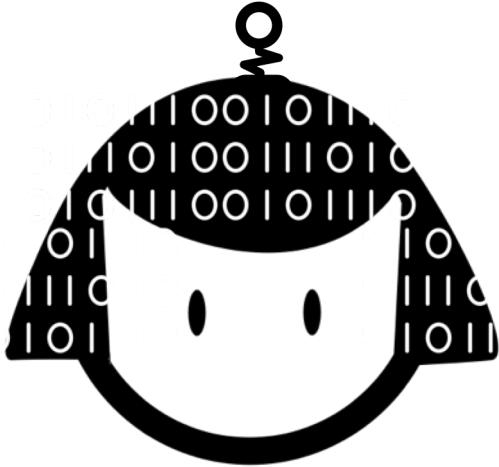


A good human conversation is
synchronous

You **wait** for a response

You **expect** a response
immediately

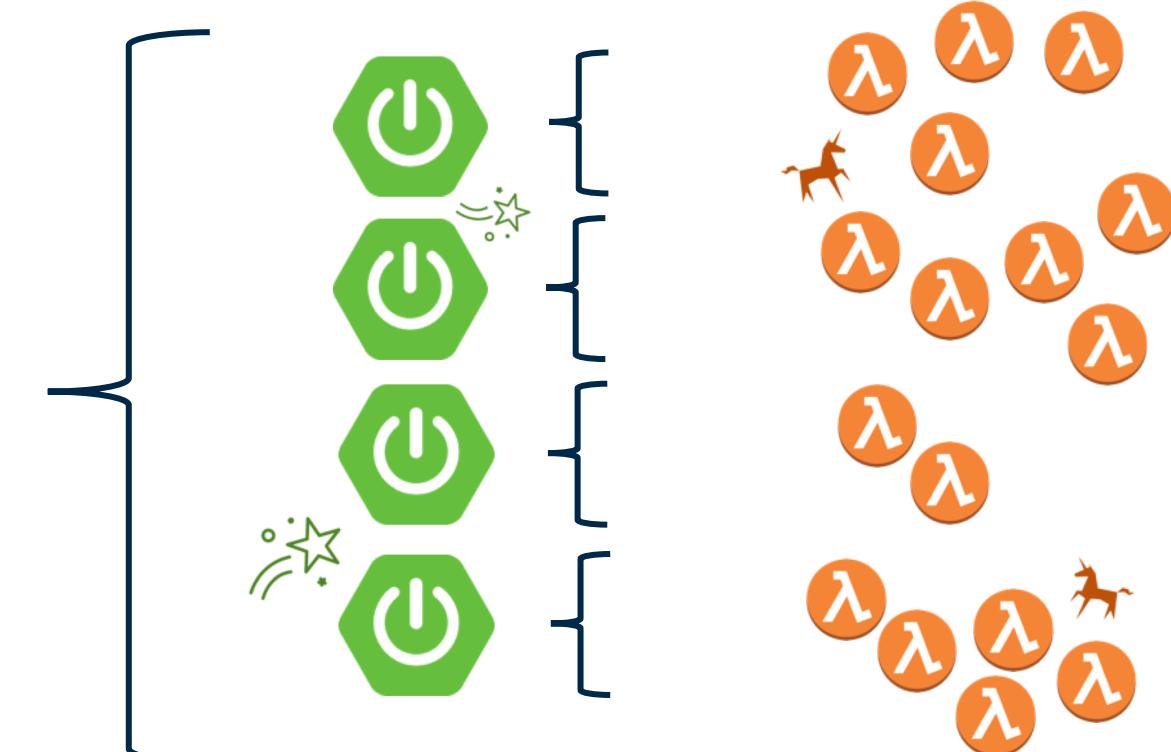
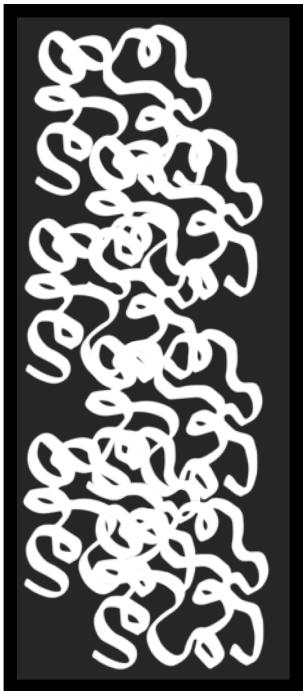




A Conversational Architecture needs to

- Be **fast!**
- **Appear synchronous** to the user (there should be a response)

This is particularly critical for a voice-based bot



Monolith



Microservices

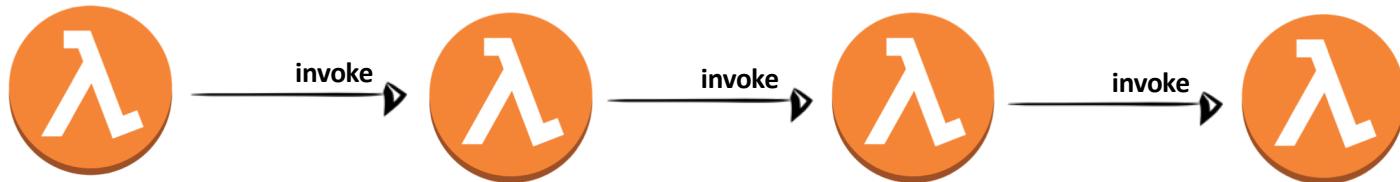


Function as a Service

**Do not just lift and shift your
microservices into FaaS**



Do not end up with a collection of distributed microservices



Combining the time to **invoke** and the time to '**spin up**' can hit performance and add complexity to your system.

Choose to 'keep warm' only as a last resort.

Look for where you can pull functionality
into reusable **libraries** instead

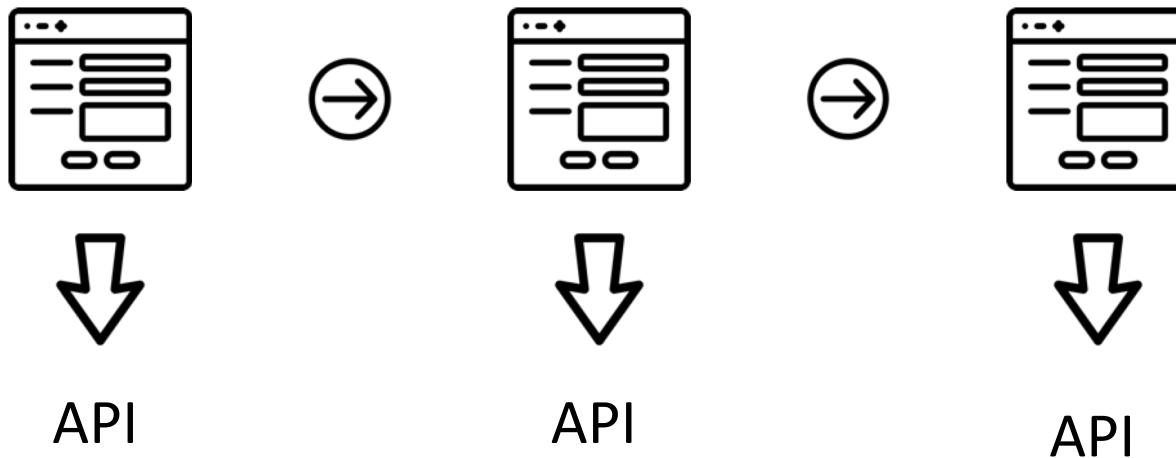


*Shhhhhh..... It's ok to have more than one
function in your Function as a Service...*

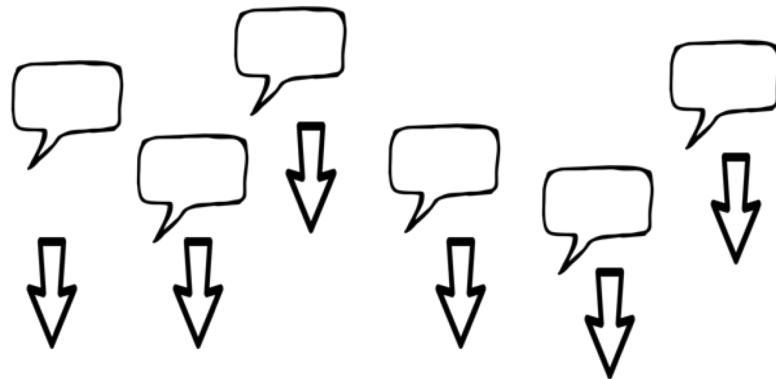
Having said that... **don't just lift and shift**
your microservices into FaaS



APIs designed for webforms tend to be set up to collect **sets of data** page by page



A Conversation is **non-linear** – the user can give **partial information**, or information that would have been **collected later** on your webform.

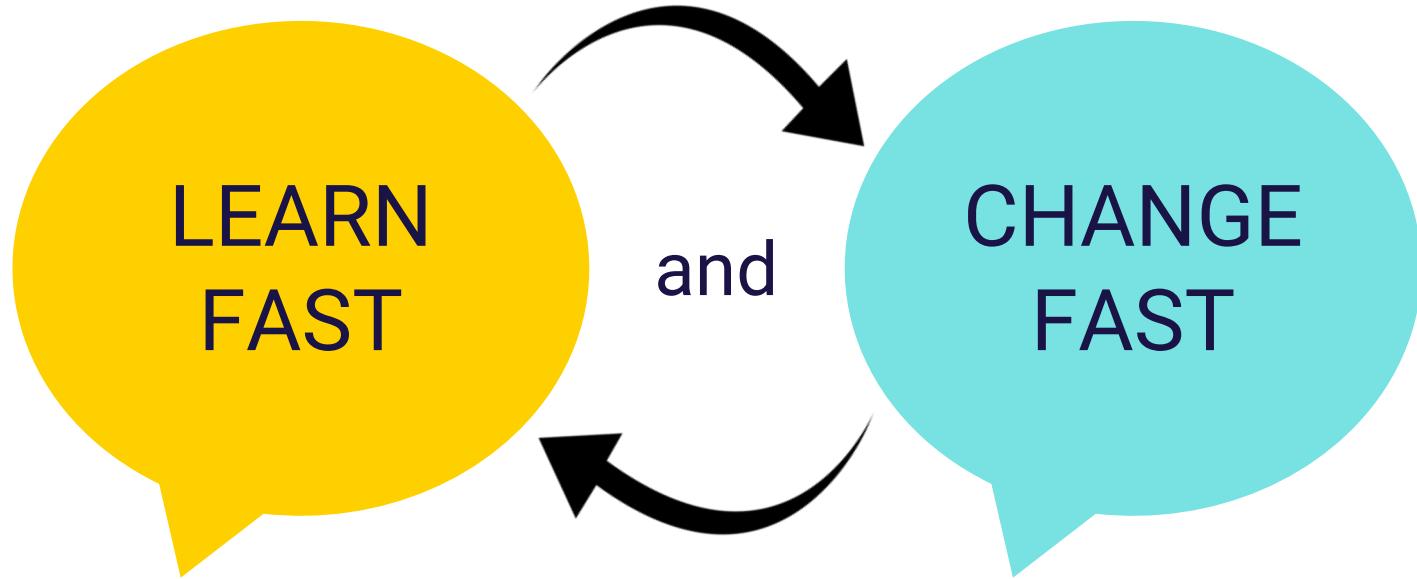


API

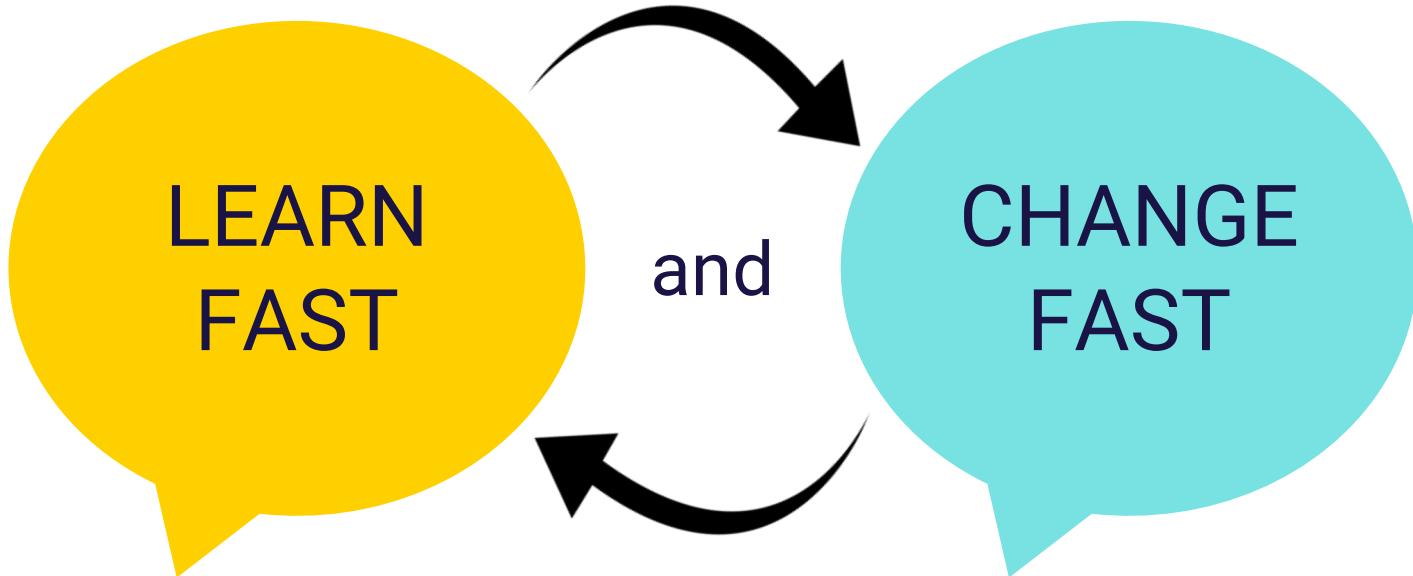
REWRITE is not a
four-letter word

**Both Serverless
and Chatbots
require new ways
of thinking about
your architecture.**

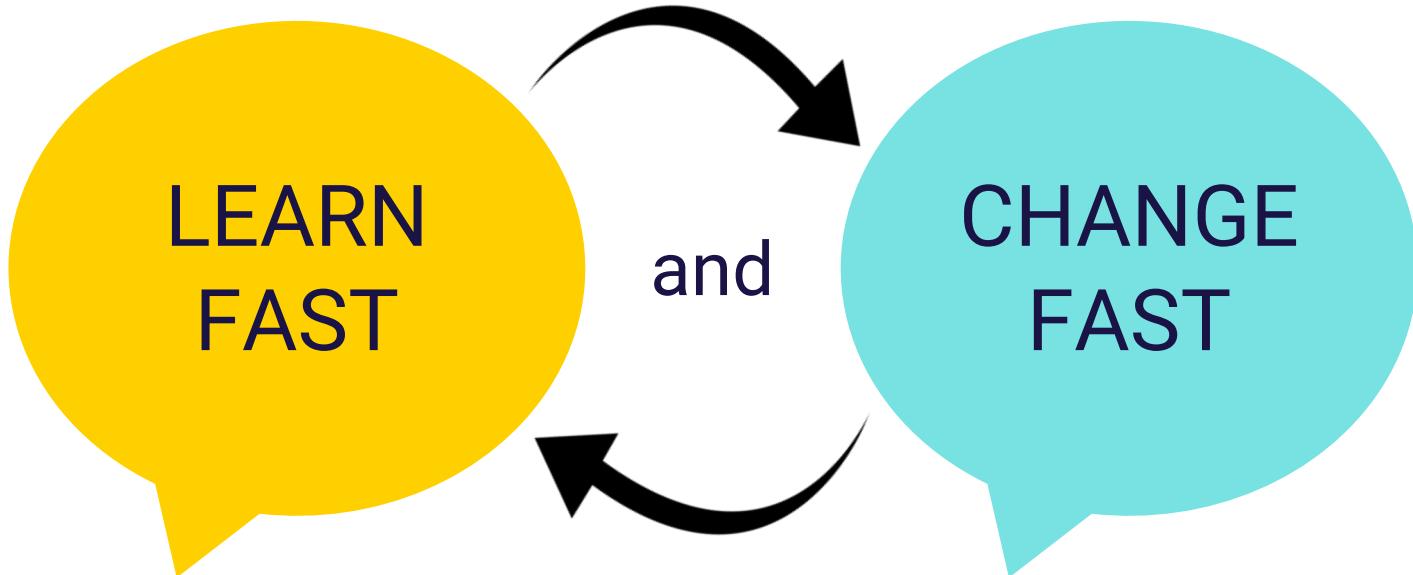
We get there **iteratively** by being able to



Both Chatbots and Serverless let us



Both Chatbots and Serverless let us



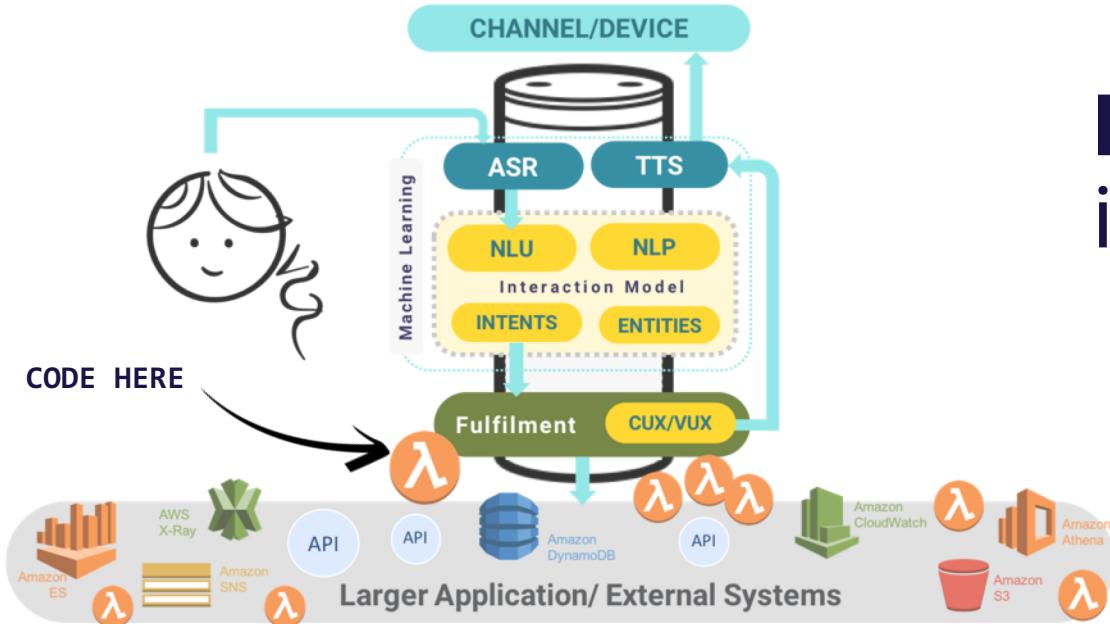
*but **Good Supporting Engineering Practices** are
needed to do both **with confidence** and **at scale***

Amazon Lex Console

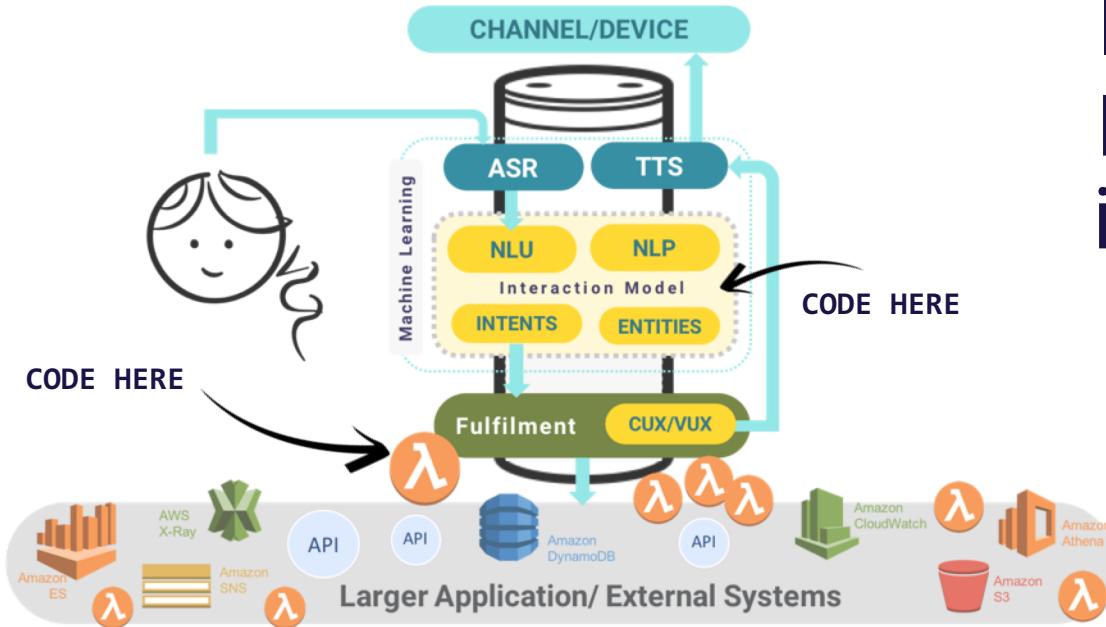
The screenshot shows the Amazon Lex console interface for the 'DemoAssistantBot' bot. On the left, there's a sidebar with sections for Intents, Slot types, and Error Handling. The main area displays the 'demo_BookMeetingIntent' intent, which has two sample utterances: 'please book a meeting at [time]', 'I want to book a meeting', and 'I need a meeting booked'. Below these, a list of other intents is shown: demo_BookMeetingIntent, demo_CalendarAvailability, demo_GenericHelpIntent, demo_GoodbyeIntent, demo_GreetingIntent, and demo_OrderDrink. The right side of the screen shows a testing interface with a text input field and a large button labeled 'Test bot (Latest)'. A message says 'Ready. Build complete.' and 'You're now ready for testing. Type an utterance below to begin conversation with your chatbot.' At the bottom, there's an 'Inspect response' section and a note about fulfillment states.

What the Web Tutorials told me building a chatbot looked like...

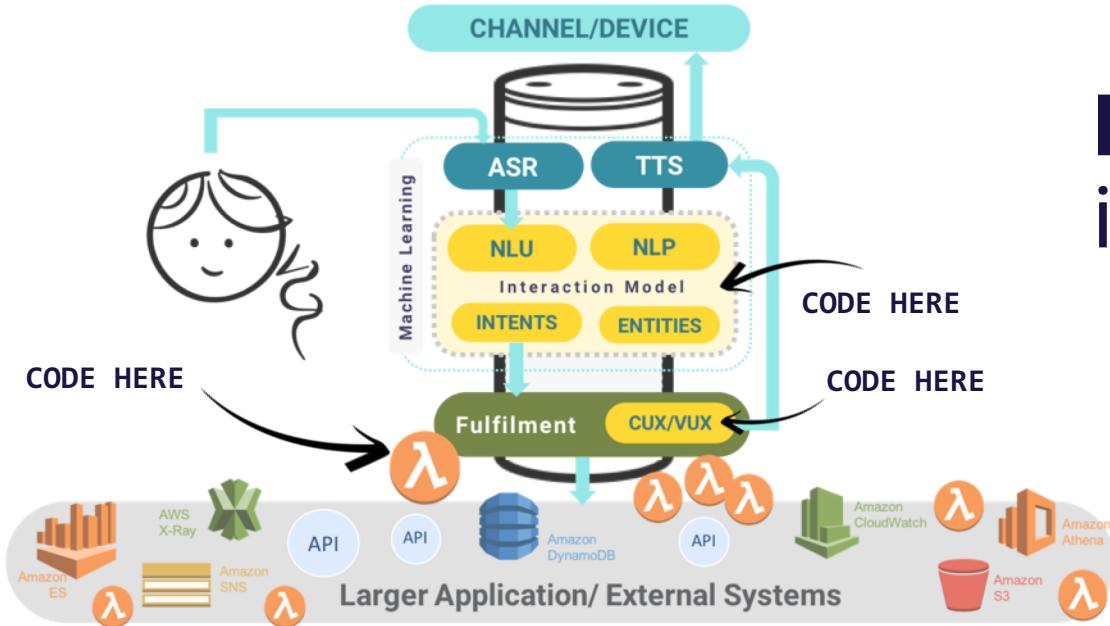
Fulfillment in Code



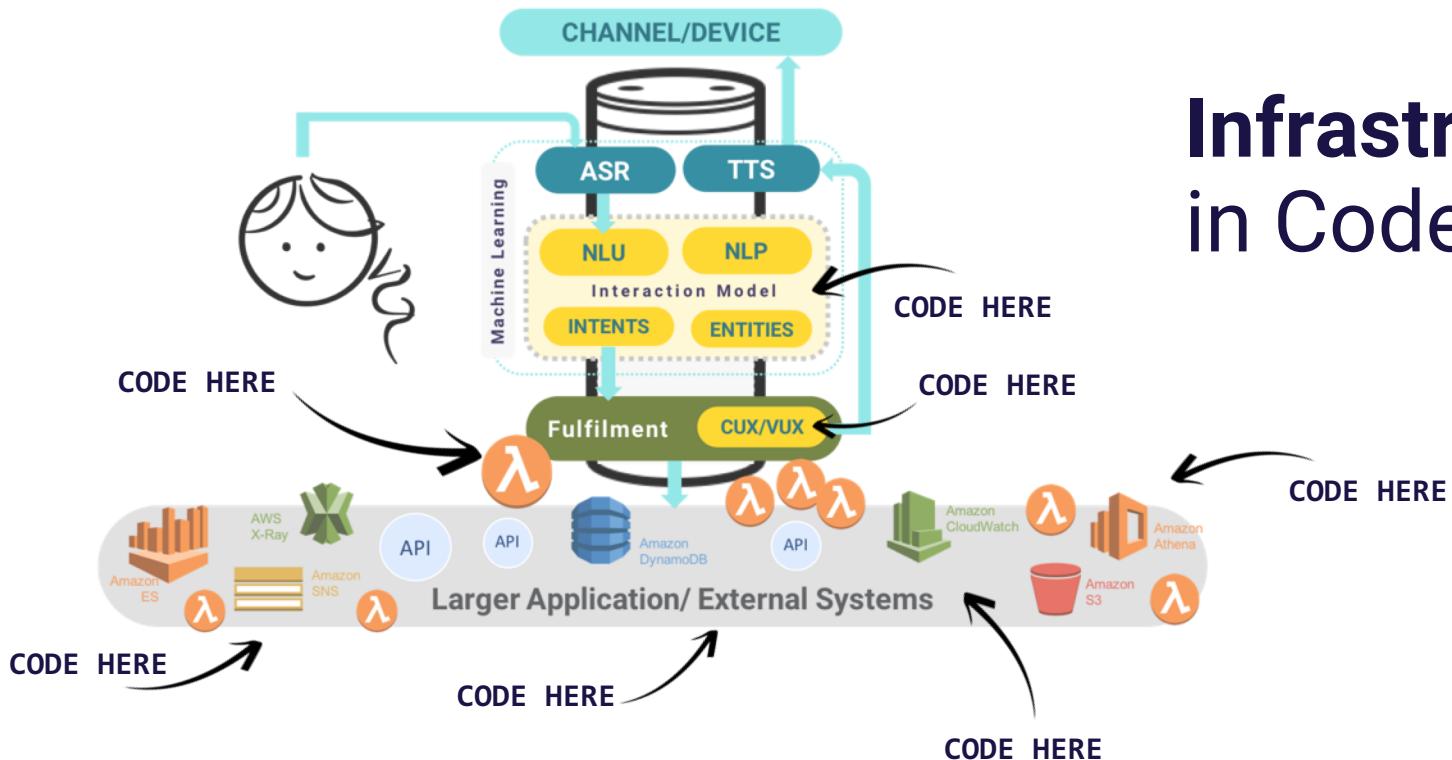
Bot Interaction Model in Code



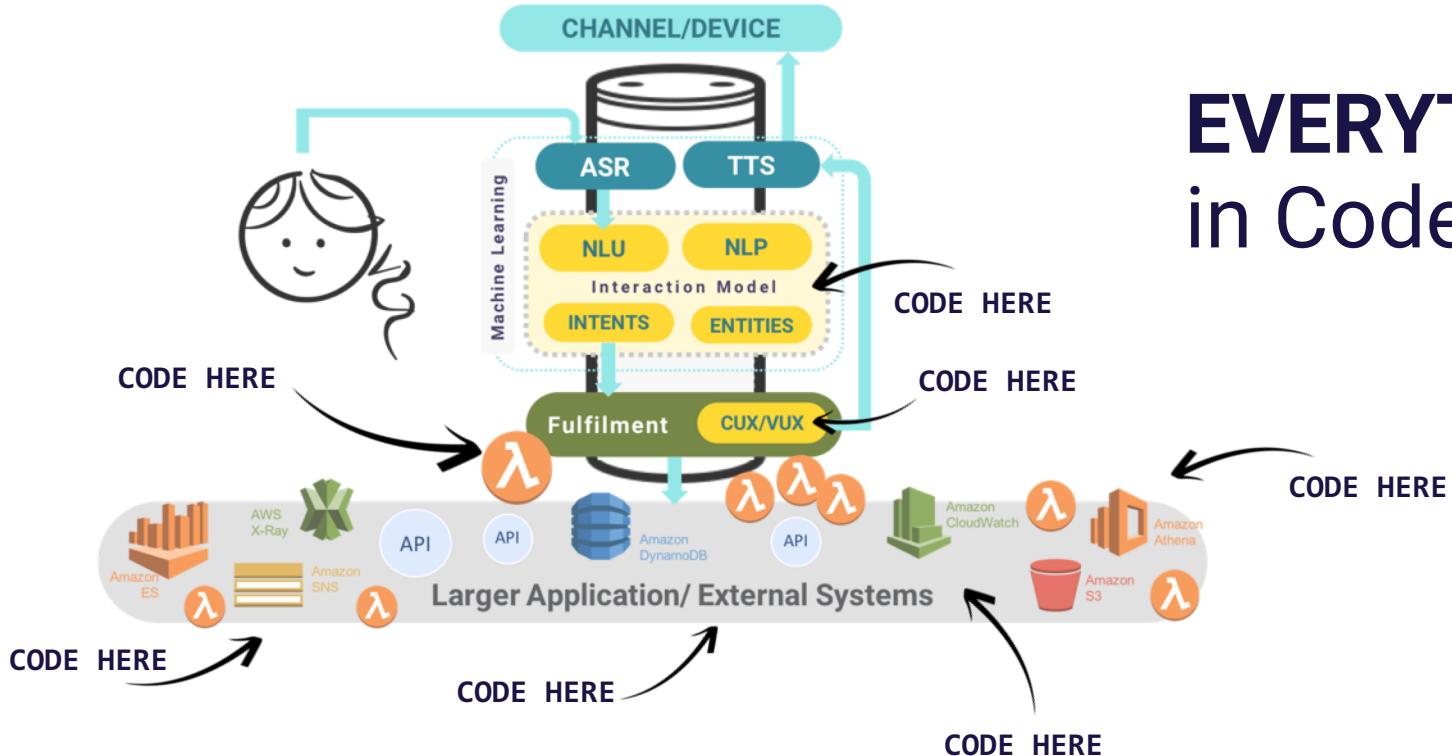
Design in Code



Infrastructure in Code

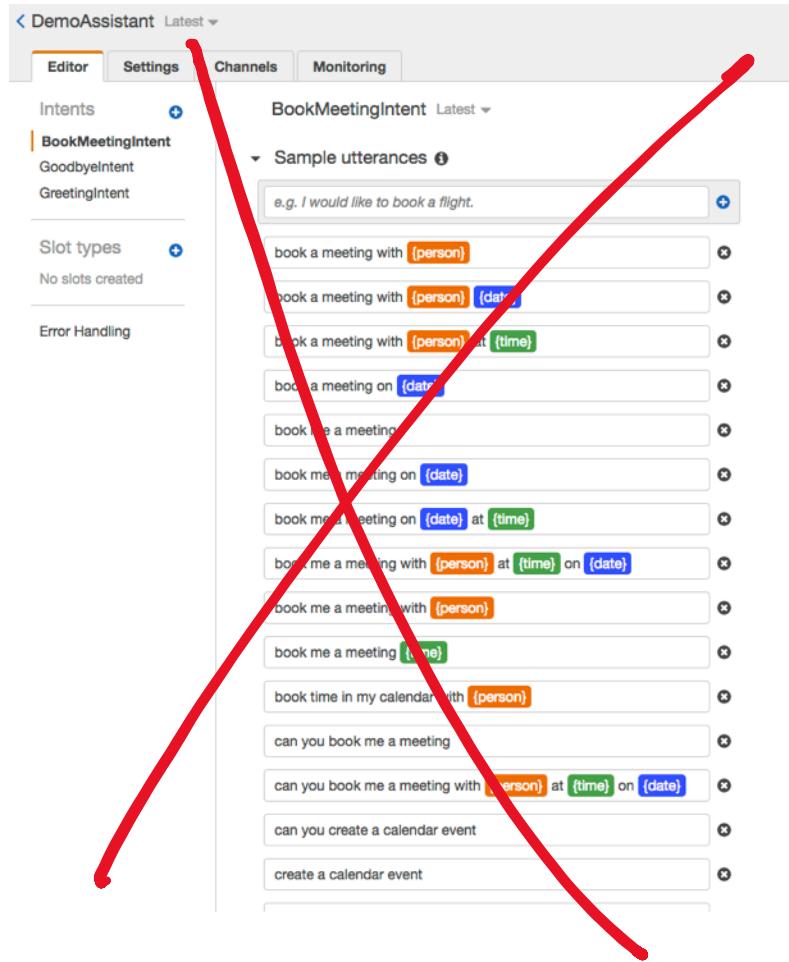


EVERYTHING in Code



- Testable
- Traceable
- Repeatable
- Observable

EVERYTHING
in Code

A screenshot of the Amazon Lex console. The left sidebar shows intents: BookMeetingIntent (selected), GoodbyeIntent, GreetingIntent. Under Slot types, it says "No slots created". Under Error Handling, there's a link. The main area shows the "BookMeetingIntent Latest" intent. A large red 'X' is drawn across the entire interface.

BookMeetingIntent Latest

Intents

- BookMeetingIntent
- GoodbyeIntent
- GreetingIntent

Slot types

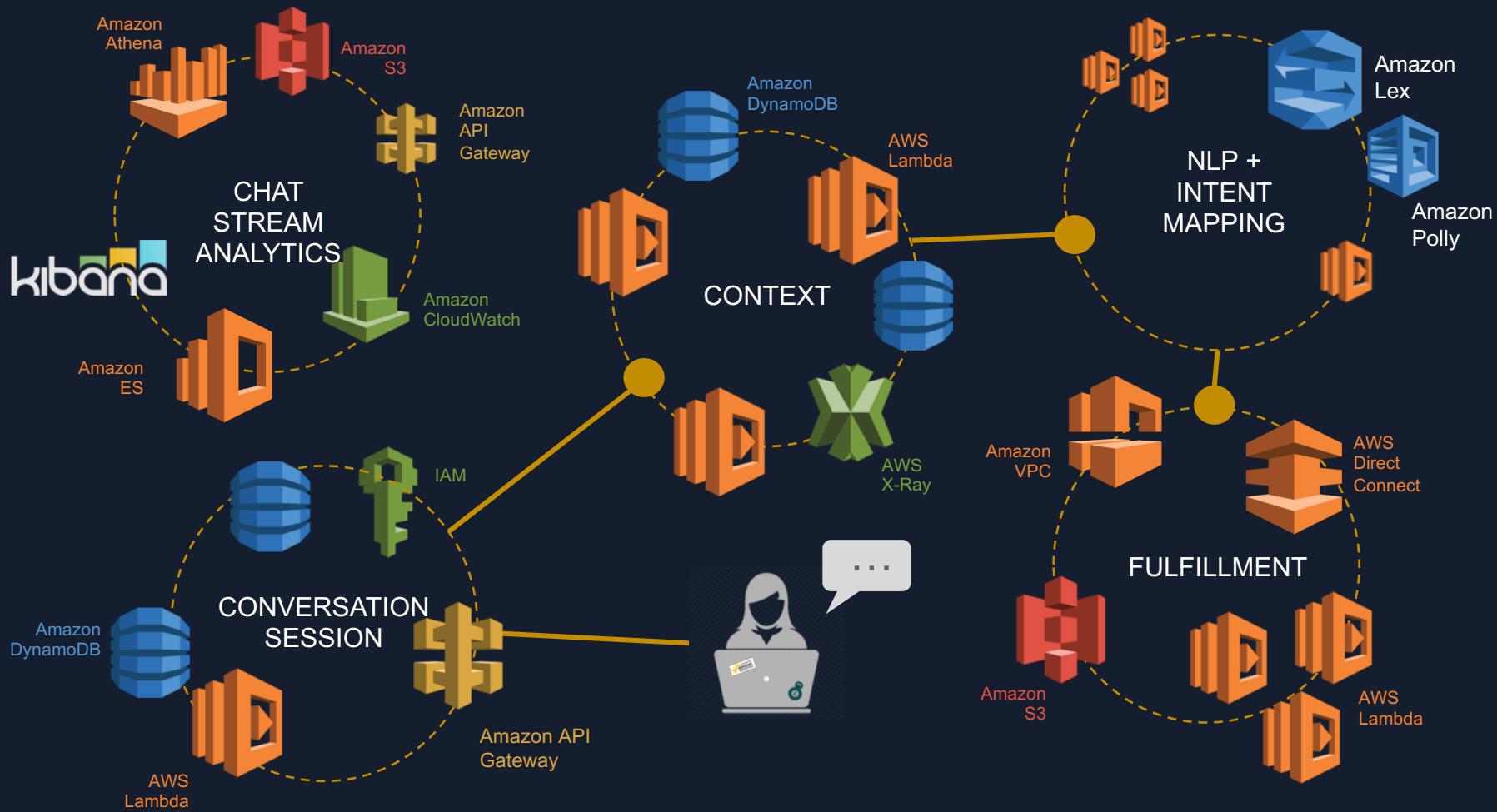
No slots created

Error Handling

Sample utterances

- e.g. I would like to book a flight.
- book a meeting with [person]
- book a meeting with [person] [date]
- book a meeting with [person] at [time]
- book a meeting on [date]
- book me a meeting
- book me a meeting on [date]
- book me a meeting with [person] at [time] on [date]
- book me a meeting with [person]
- book me a meeting [time]
- book time in my calendar with [person]
- can you book me a meeting
- can you book me a meeting with [person] at [time] on [date]
- can you create a calendar event
- create a calendar event

```
},
"sampleUtterances": [
  "book a meeting with {person}",
  "book a meeting with {person} {date}",
  "book a meeting with {person} at {time}",
  "book a meeting on {date}",
  "book me a meeting",
  "book me a meeting on {date}",
  "book me a meeting on {date} at {time}",
  "book me a meeting with {person} at {time} on {date}",
  "book me a meeting with {person}",
  "book me a meeting {time}",
  "book time in my calendar with {person}",
  "can you book me a meeting",
  "can you book me a meeting with {person} at {time} on {date}",
  "can you create a calendar event",
  "create a calendar event",
  "i need a meeting booked",
  "i want to book a meeting",
  "please book a meeting at {time}"
],
"slots": [
  {
    "description": "The person the meeting is with",
    "name": "person",
    "priority": 1,
    "slotConstraint": "Required",
    "slotType": "AMAZON.Person",
    "valueElicitationPrompt": {
      "maxAttempts": 5,
      "messages": [
        {
          "content": "Who would you like to meet with?",
          "contentType": "PlainText"
        }
      ]
    },
    "description": "The date of the meeting",
    "name": "date",
    "priority": 2,
    "slotConstraint": "Required",
    "slotType": "AMAZON.DATE",
    "valueElicitationPrompt": {
      "maxAttempts": 2,
      "messages": [
        {
          "content": "What date would you like the meeting on?",
          "contentType": "PlainText"
        },
        {
          "content": "Sure! What date would you like to meet on?",
          "contentType": "PlainText"
        }
      ]
    },
    "description": "The time of the meeting",
    "name": "time",
    "priority": 3,
    "slotConstraint": "Required",
    "slotType": "AMAZON.TTME"
  }
],
```



THE DIGITAL ASSISTANT BOT PLATFORM ARCHITECTURE

Conversation as Code - Amazon Lex

Fully Validated and Repeatable Bot Build and Deploy to any AWS Account



Everything written in CFN,
with JSON for API calls to
Amazon Lex Model Building
API as part of deploy

Conversation as Code - Amazon Lex

Fully Validated and Repeatable Bot Build and Deploy to any AWS Account



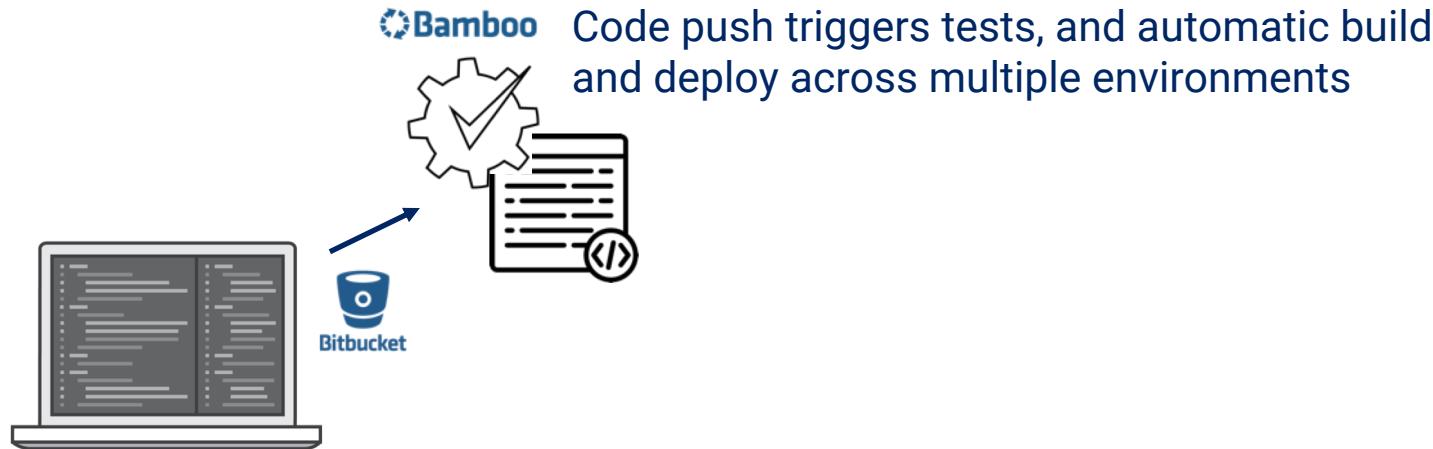
Everything written in CFN,
with JSON for API calls to
Amazon Lex Model Building
API as part of deploy

- Unit Test
 - Lambdas
 - Libraries
- Static analysis
 - Cloud Formation
 - Code
 - JSON for API calls

Local and Pre-Commit
Hook Tests

Conversation as Code - Amazon Lex

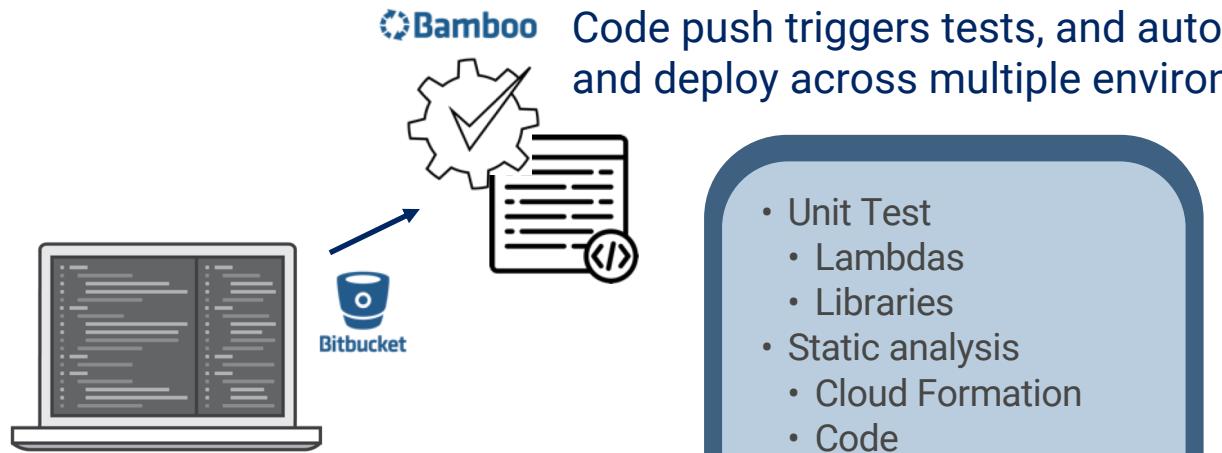
Fully Validated and Repeatable Bot Build and Deploy to any AWS Account



Everything written in CFN,
with JSON for API calls to
Amazon Lex Model Building
API as part of deploy

Conversation as Code - Amazon Lex

Fully Validated and Repeatable Bot Build and Deploy to any AWS Account



Everything written in CFN,
with JSON for API calls to
Amazon Lex Model Building
API as part of deploy

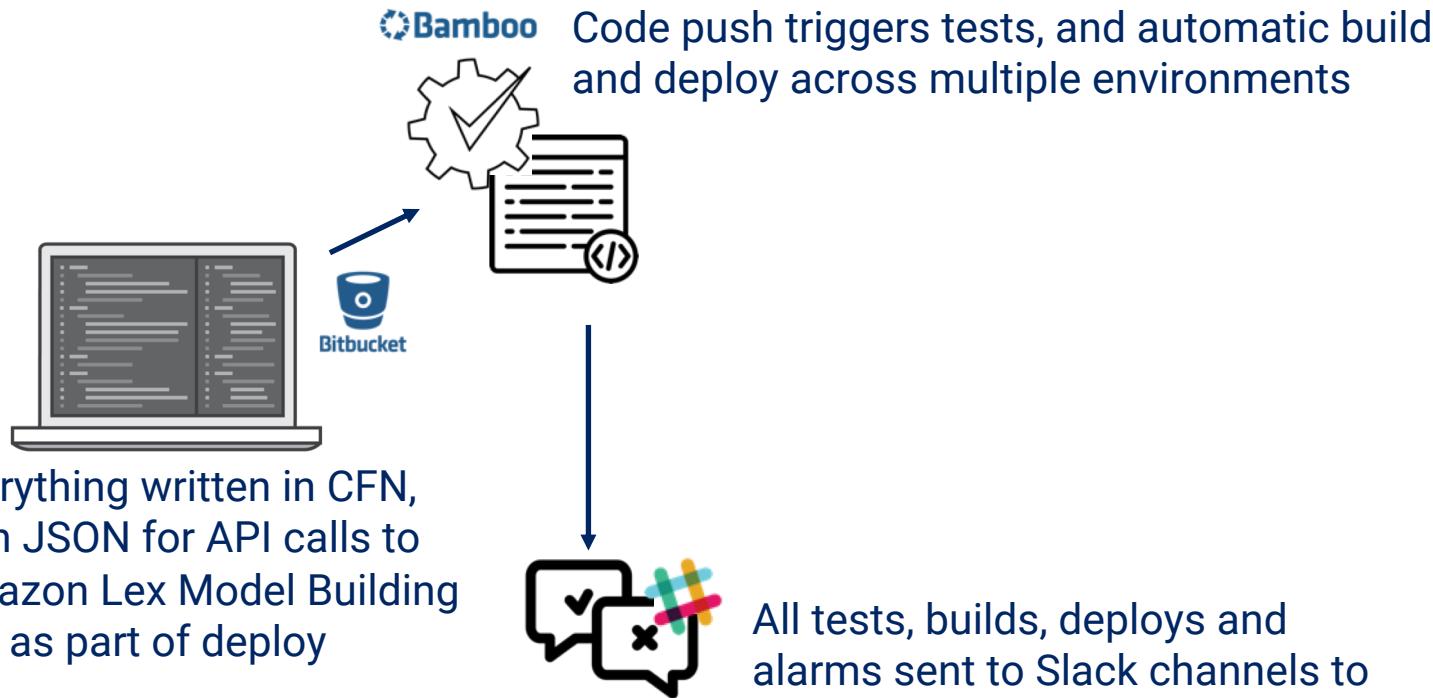
Bamboo Code push triggers tests, and automatic build and deploy across multiple environments

- Unit Test
 - Lambdas
 - Libraries
- Static analysis
 - Cloud Formation
 - Code
- JSON for API calls

Build Tests

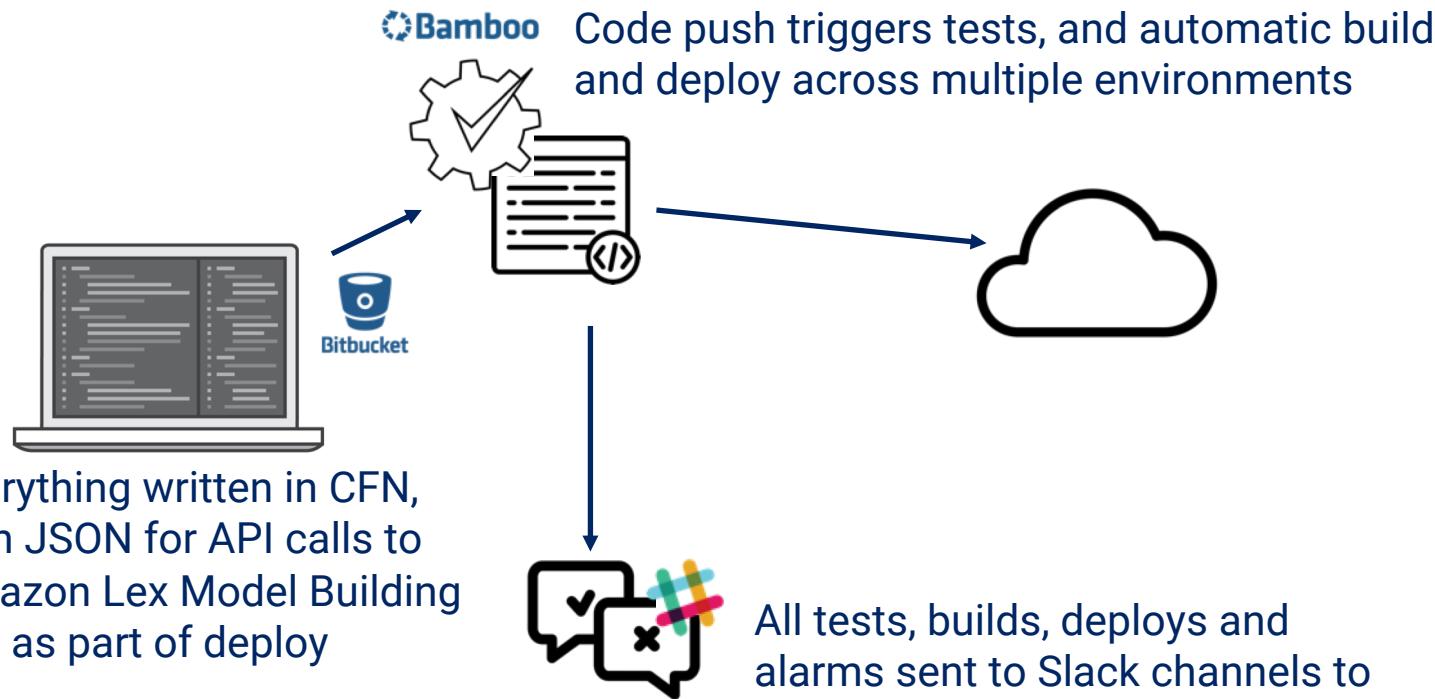
Conversation as Code - Amazon Lex

Fully Validated and Repeatable Bot Build and Deploy to any AWS Account



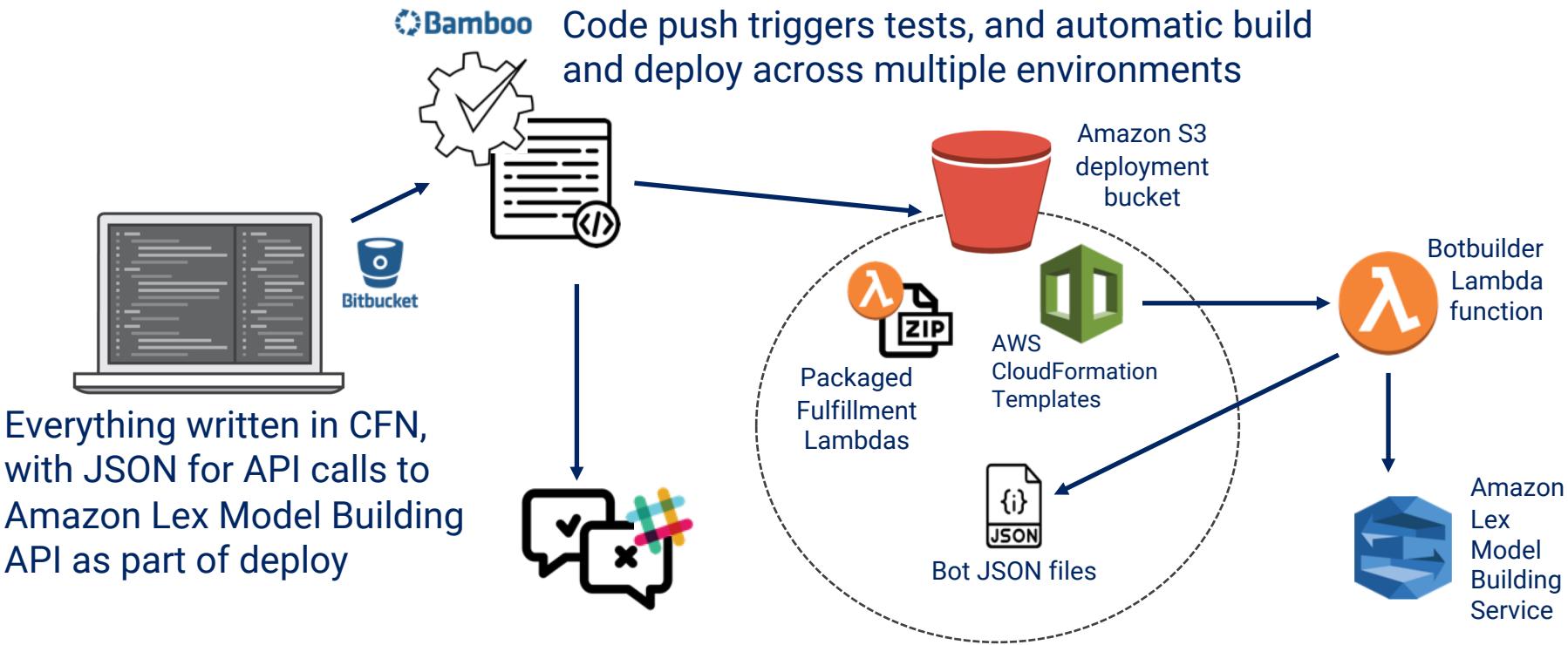
Conversation as Code - Amazon Lex

Fully Validated and Repeatable Bot Build and Deploy to any AWS Account



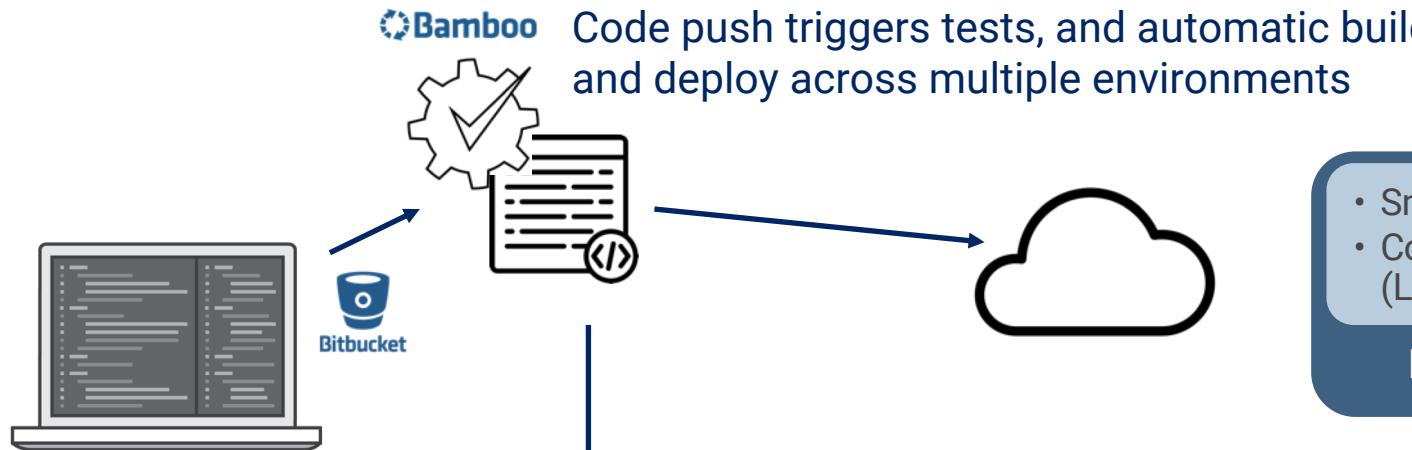
Conversation as Code - Amazon Lex

Fully Validated and Repeatable Bot Build and Deploy to any AWS Account



Conversation as Code - Amazon Lex

Fully Validated and Repeatable Bot Build and Deploy to any AWS Account



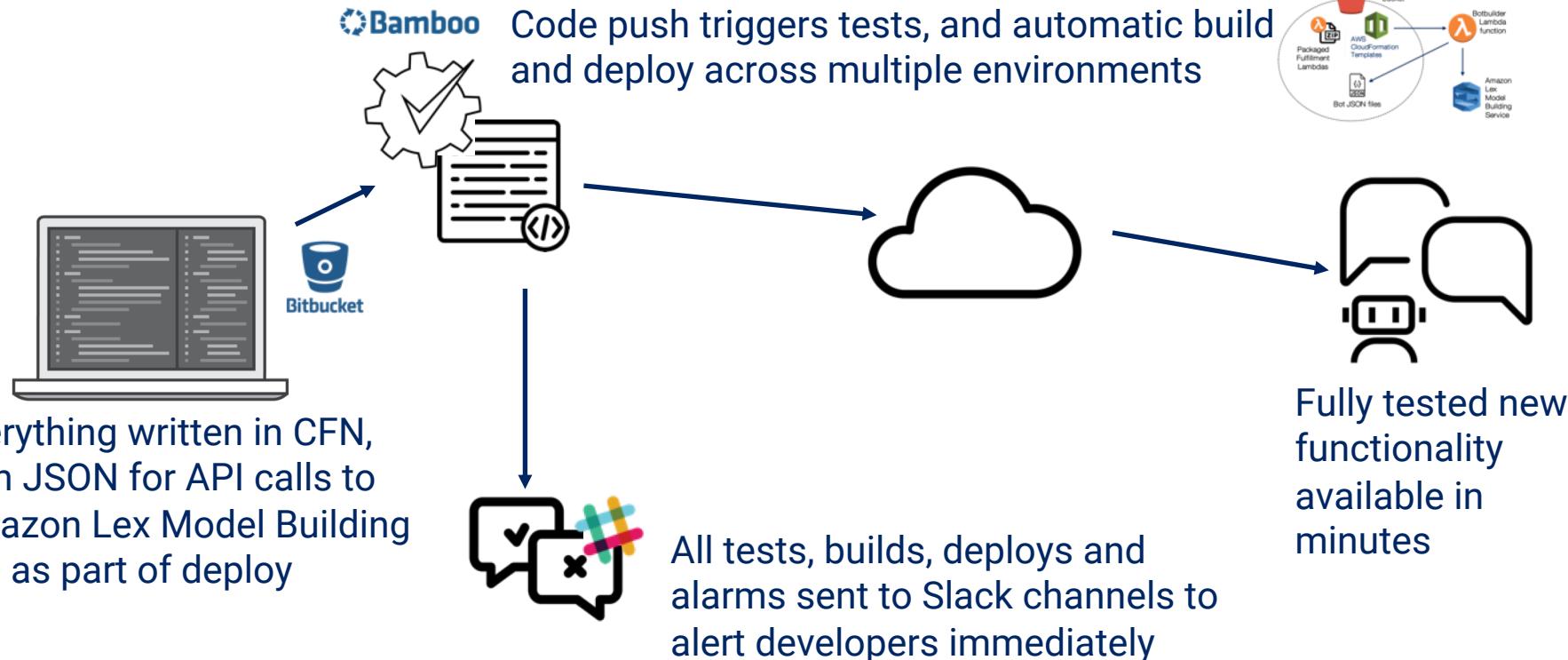
- Smoke Tests
- Component Tests (Lex) Non-Prod Only

Deploy Tests

All tests, builds, deploys and alarms sent to Slack channels to alert developers immediately

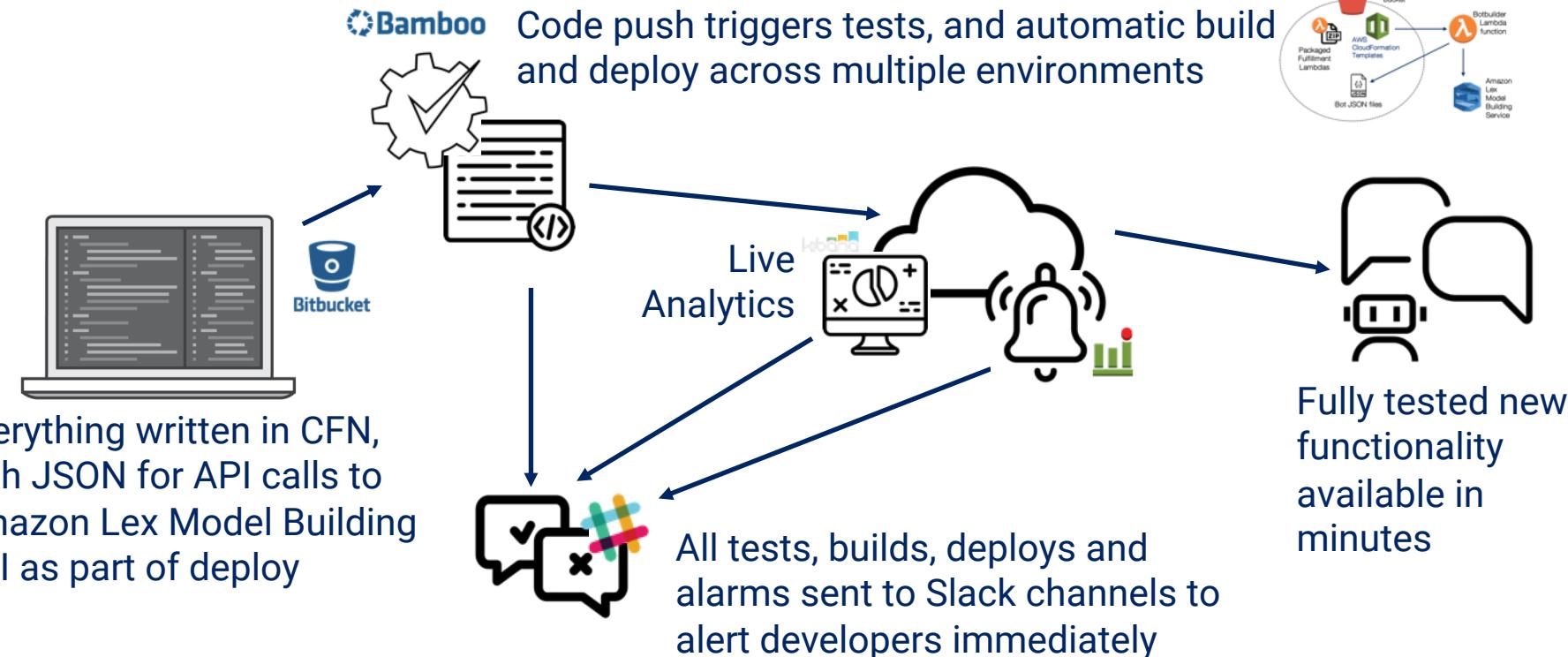
Conversation as Code - Amazon Lex

Fully Validated and Repeatable Bot Build and Deploy to any AWS Account



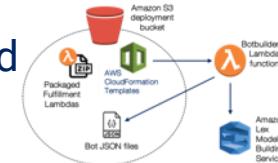
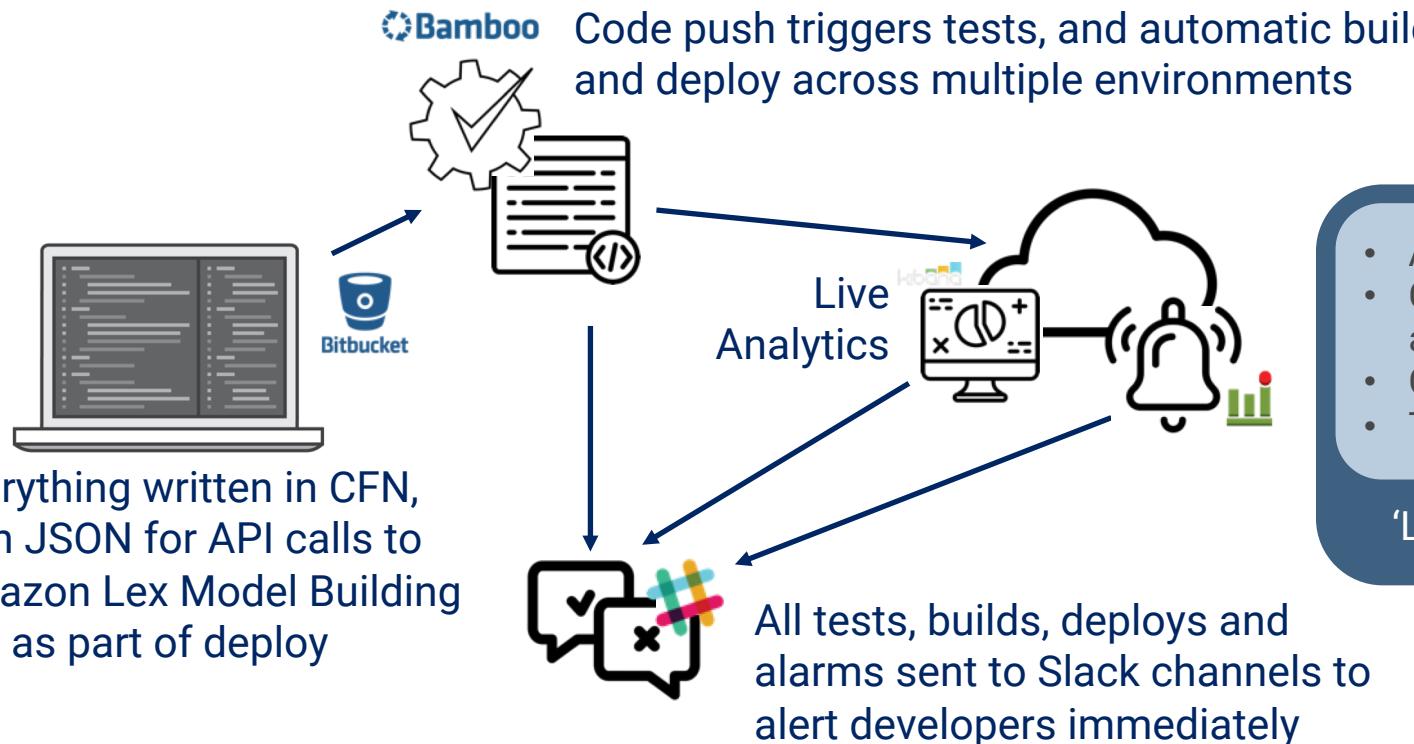
Conversation as Code - Amazon Lex

Fully Validated and Repeatable Bot Build and Deploy to any AWS Account



Conversation as Code - Amazon Lex

Fully Validated and Repeatable Bot Build and Deploy to any AWS Account

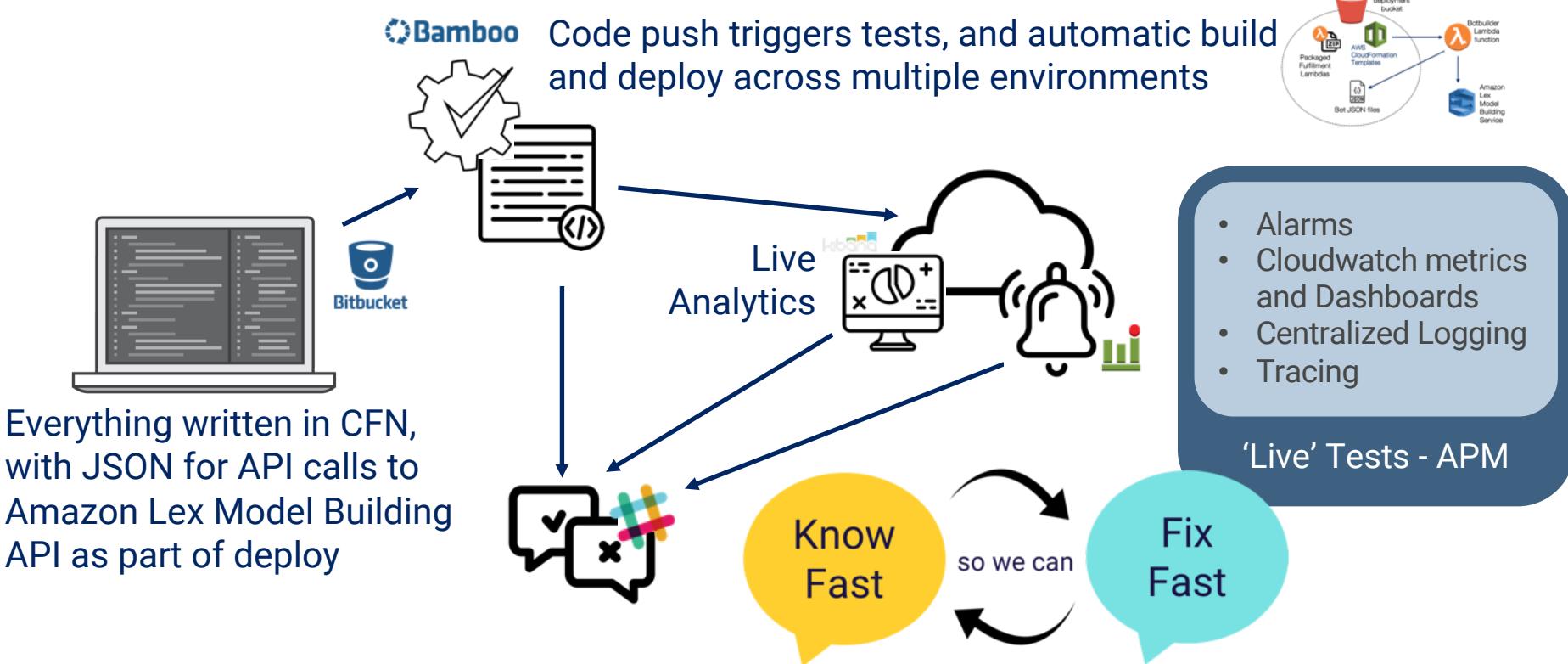


- Alarms
- Cloudwatch metrics and Dashboards
- Centralized Logging
- Tracing

'Live' Tests - APM

Conversation as Code - Amazon Lex

Fully Validated and Repeatable Bot Build and Deploy to any AWS Account



Monitor, no, seriously, monitor



If your **chatbot stopped working**, would you **know**? Don't just wait for user complaints!

Know what isn't working and **fix** it, and keep **adding more** of what is!

Conversational Analytics are vital



Work out what **metrics** you need to track to know how your chatbot is performing.

Know what isn't working and **fix** it, and keep **adding more** of what is!

Getting **Feedback** is vital



You won't get it right first time – **listen** to your users.

Know what isn't working and **fix** it, and keep **adding more** of what is!

Being able to

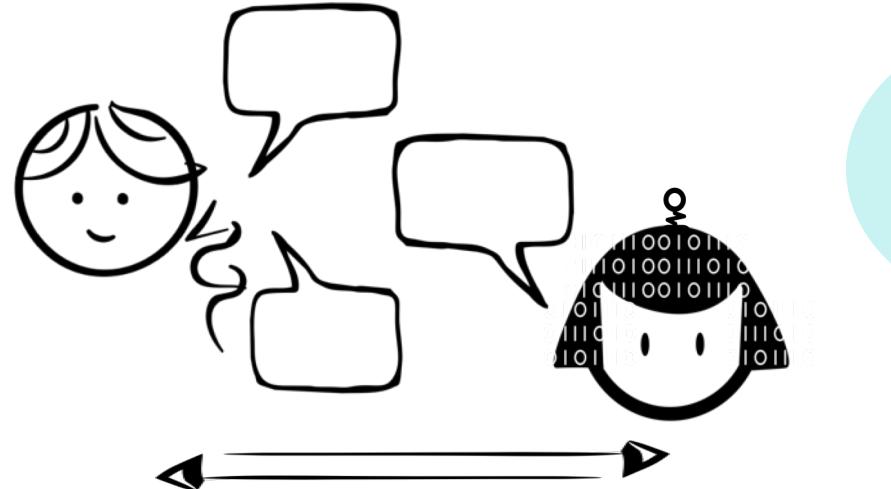
Change
Fast

lets us

Learn
Fast



LEARN
FAST



CHANGE
FAST

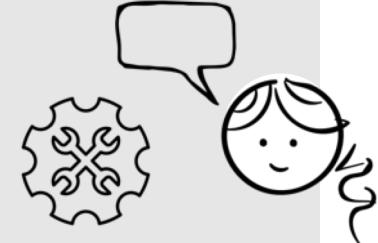
Conversational UI is about
Listening to what the user
wants to do...

...and that's the secret to a **great chatbot!**

KEY TAKEAWAYS



- Model your Chatbot UI and Architecture based on observing Humans and real Conversation.
Break out of old mindsets!
- New tech still needs the good engineering practices you already know



Got questions?
Let's chat!



@virtualgill

Appendix – Additional Information on Conversational Design

Starting out:

<https://hackernoon.com/new-to-conversational-design-start-here-7f2f3a1b81bb>

Error Handling in Chatbot Series:

Part 1 — Voice Recognition Errors: The one where we end up shouting at a computer

<https://chatbotsmagazine.com/difficult-conversations-1-voice-recognition-error-handling-74d93056ddce>

Part 2 – Conversational Errors: The one where it gets really interesting

<https://chatbotsmagazine.com/helping-your-baby-bot-learn-to-chat-like-a-grown-up-bot-99f5170f1c55>

Part 3 – Technical Faults: The one we want to pretend will never happen (or ‘Help! My chatbot has fallen over and can’t get up’)

<https://chatbotsmagazine.com/good-ux-when-your-chatbot-is-having-a-very-bad-day-e4f22885d7fb>

You can find my other Blogs at <https://medium.com/@virtualgill>

Come talk to me on Twitter – I hang out there a lot @virtualgill

For my other talks and ways to get in contact with me head to <http://virtualgill.io>