

A close-up, low-angle photograph of the Statue of Liberty, showing her head, crown, and right arm holding the torch. The statue is rendered in a teal/cyan color. The background is a plain, light color.

QConSF

Chatbots and Serverless: A match made in the Cloud

Gillian Armstrong
@virtualgill



Liberty™
Information
Technology

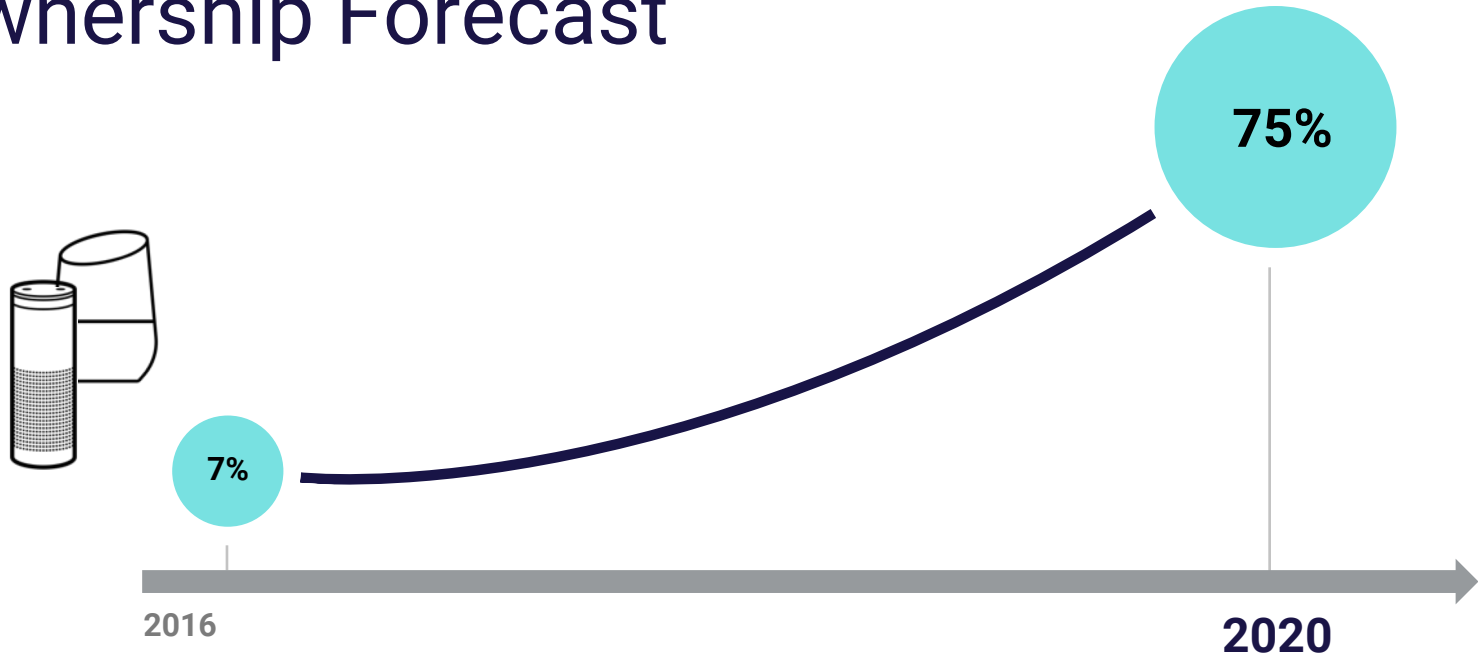


Gillian Armstrong
@virtualgill

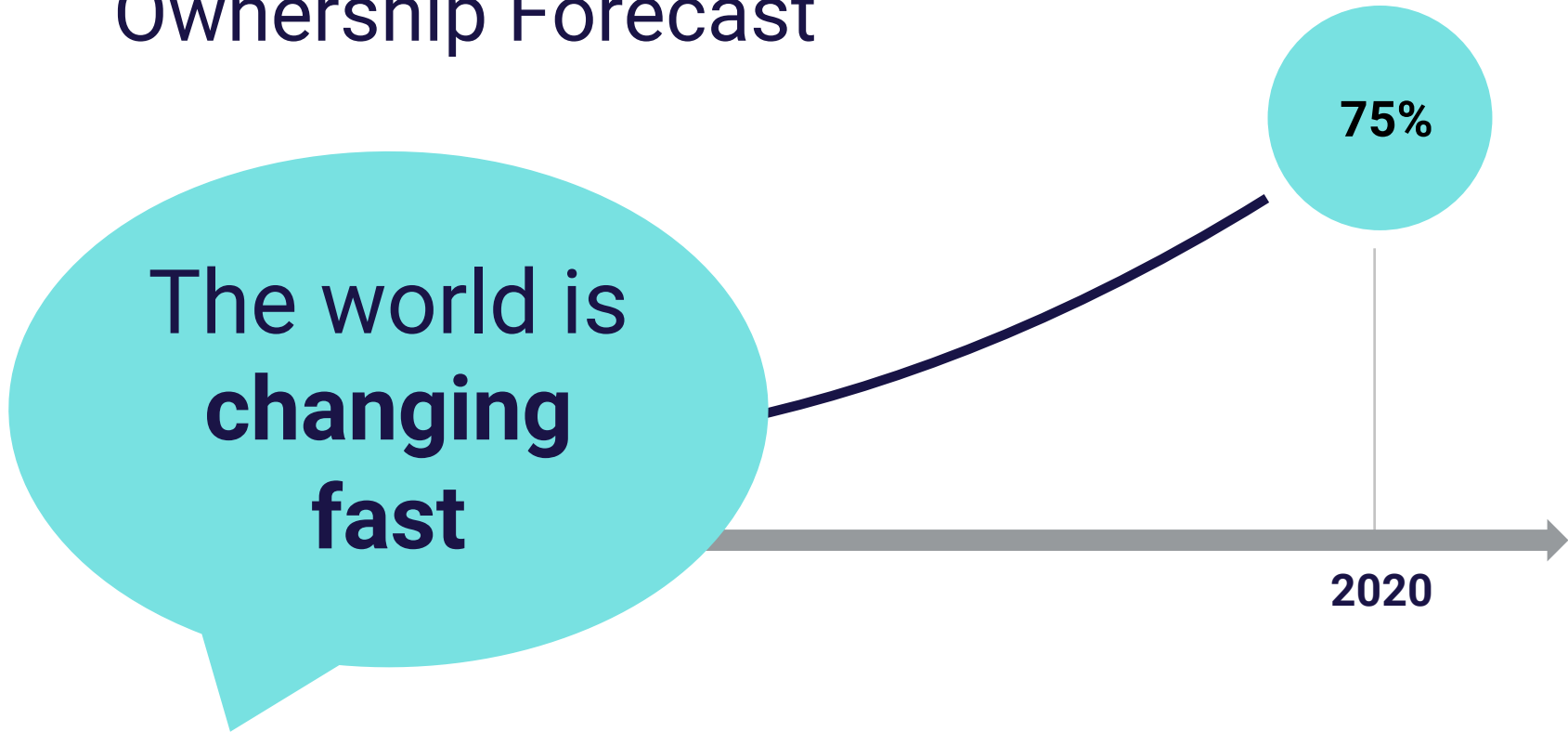
Technologist
Cognitive Technologies

Liberty IT

Smart Speaker US Household Ownership Forecast




Smart Speaker US Household Ownership Forecast

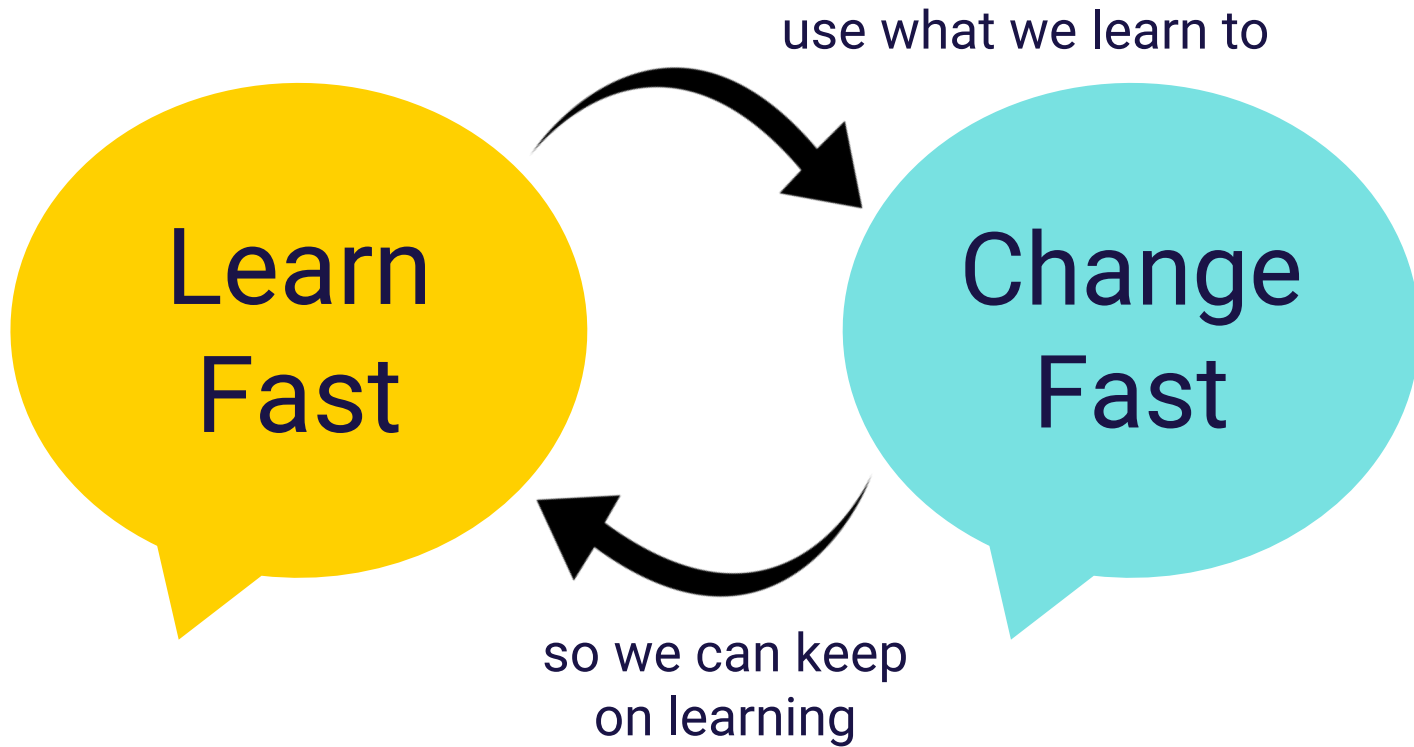




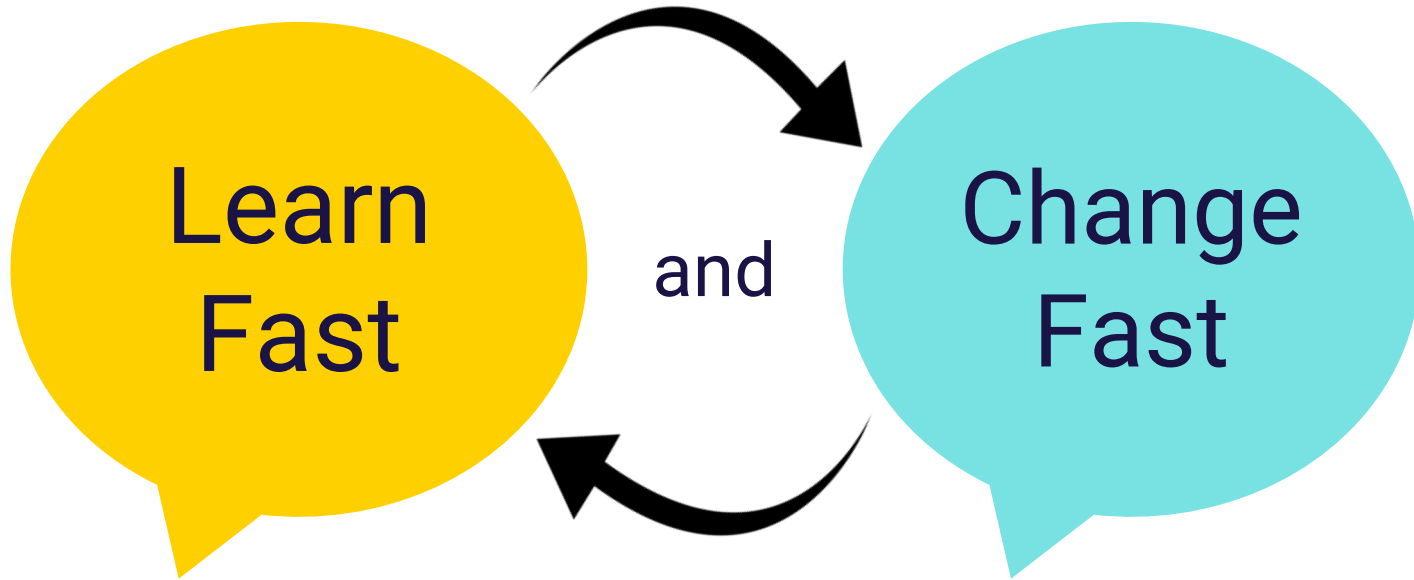
The world is
**changing
fast**



So we need to
be able to
learn fast



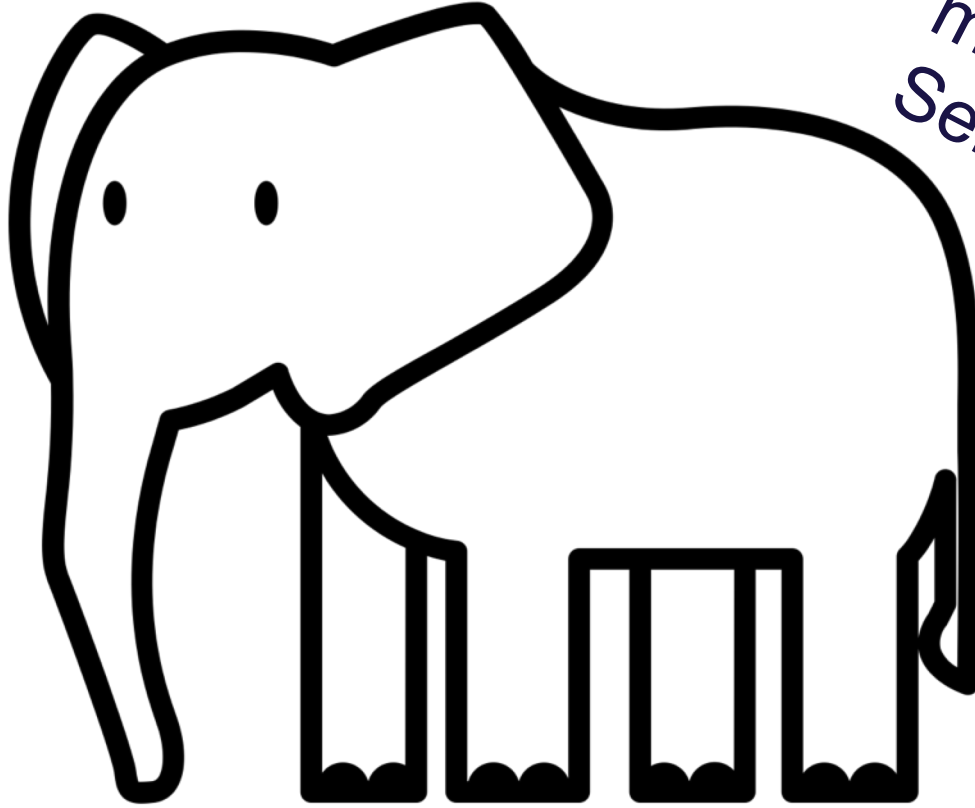
Both **Chatbots** and **Serverless** let us



What is a
Chatbot???

???

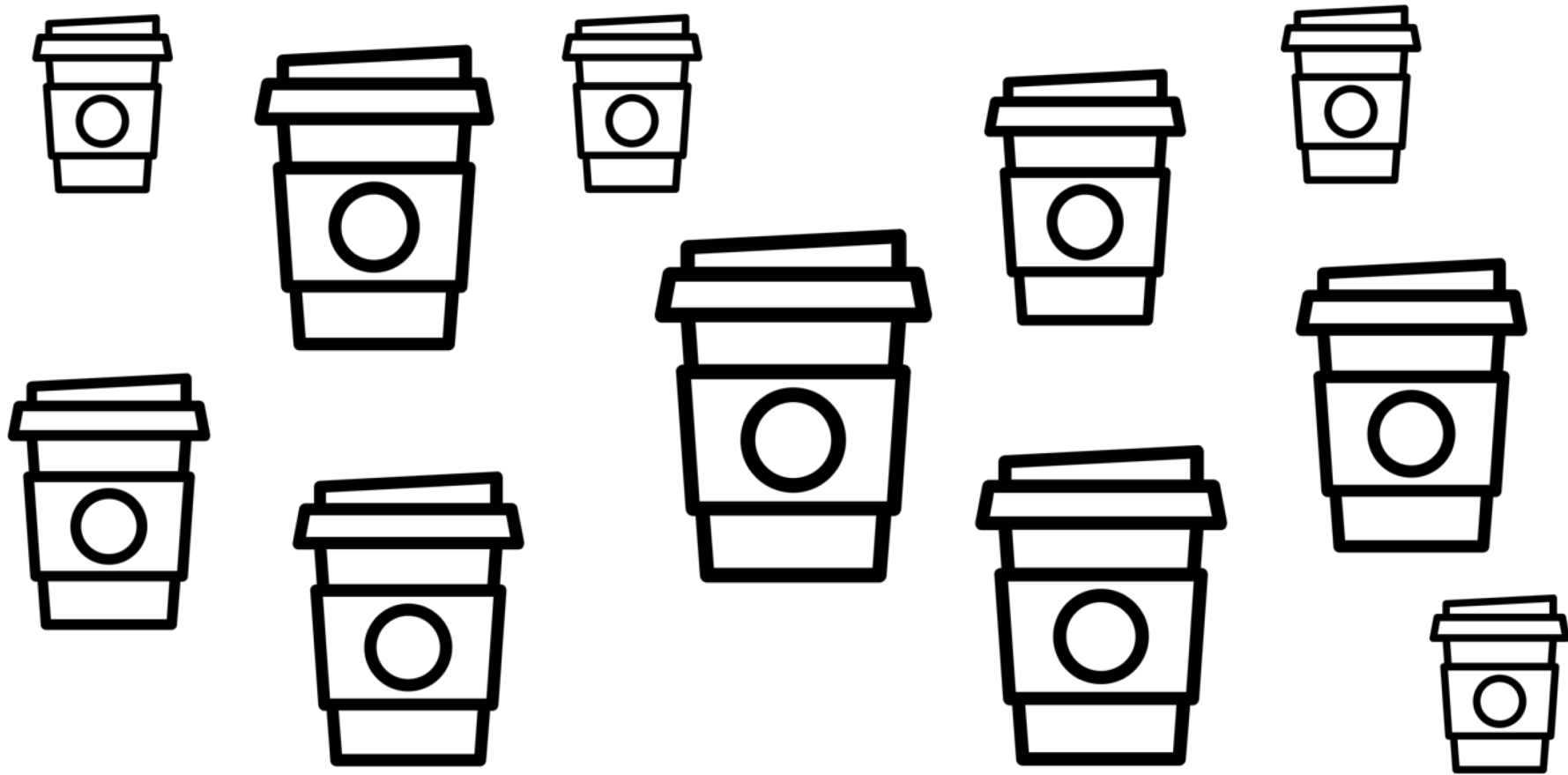
???



What do you
mean by
Serverless???

???





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

BETA

Digital Assistant

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...

Assistant

Feedback

About Me

Approvals

To Do

Notifications

About Liberty
Employee Center
SBU or Department
myConnections

View All News

What's going on?

My Places View all Activity

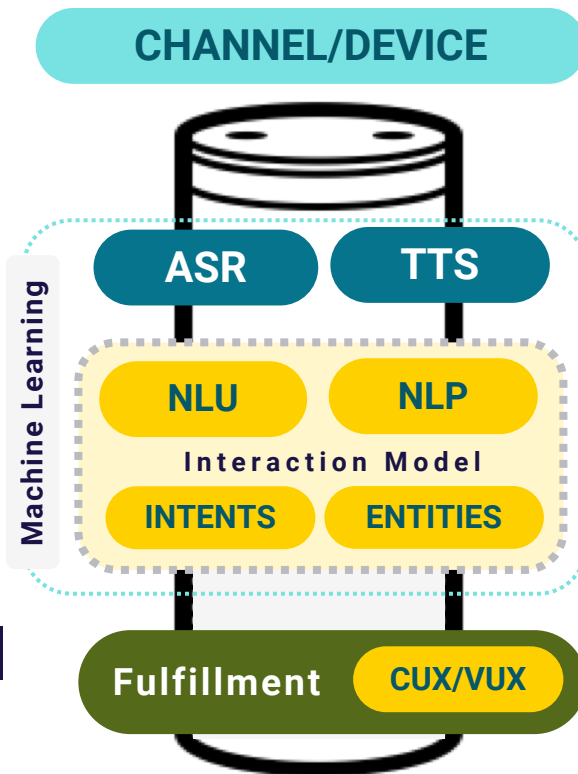
Katherine Longnameton commented on Designers + Developers = Better Together (and Happy Customers) Gain insights on methods and... in User Experience a minute ago

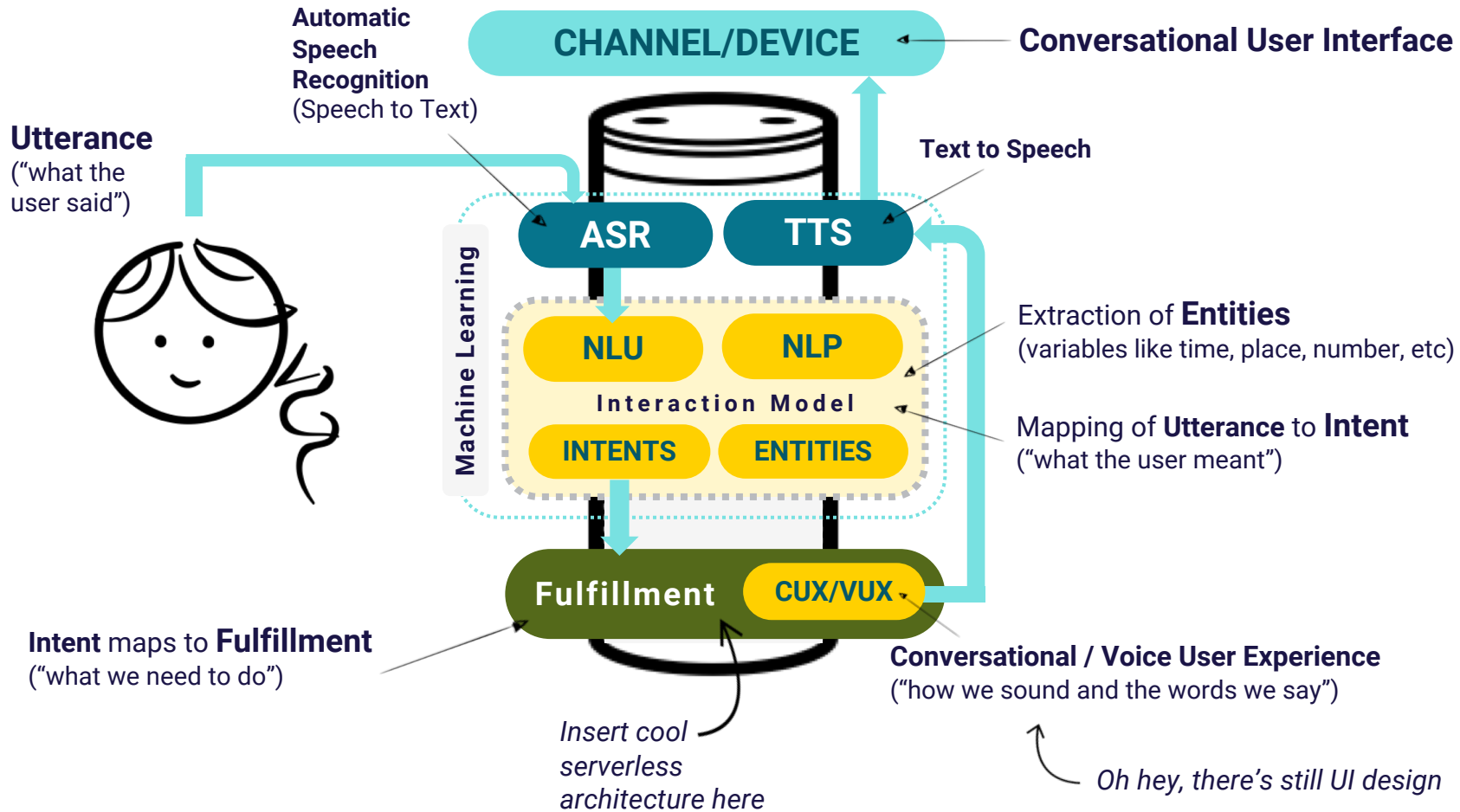
Katherine Longnameton commented on Designers + Developers = Better Together (and Happy Customers) Gain insights on methods and... in User Experience a minute ago

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

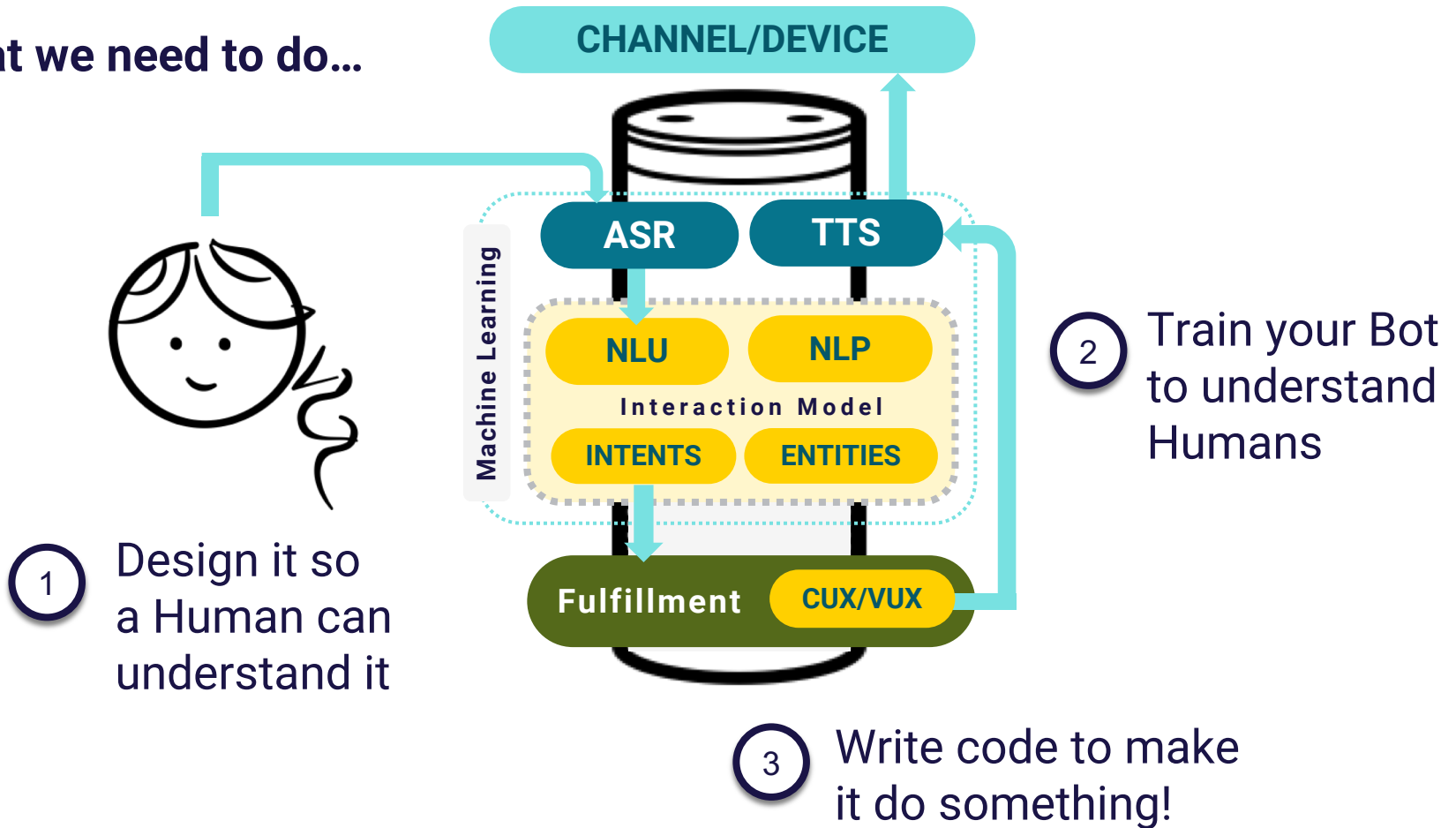
Katherine Longnameton commented on Designers + Developers = Better Together (and Happy Customers) Gain insights on methods and... in User Experience a minute ago

A very quick overview of a Chatbot using Conversational AI

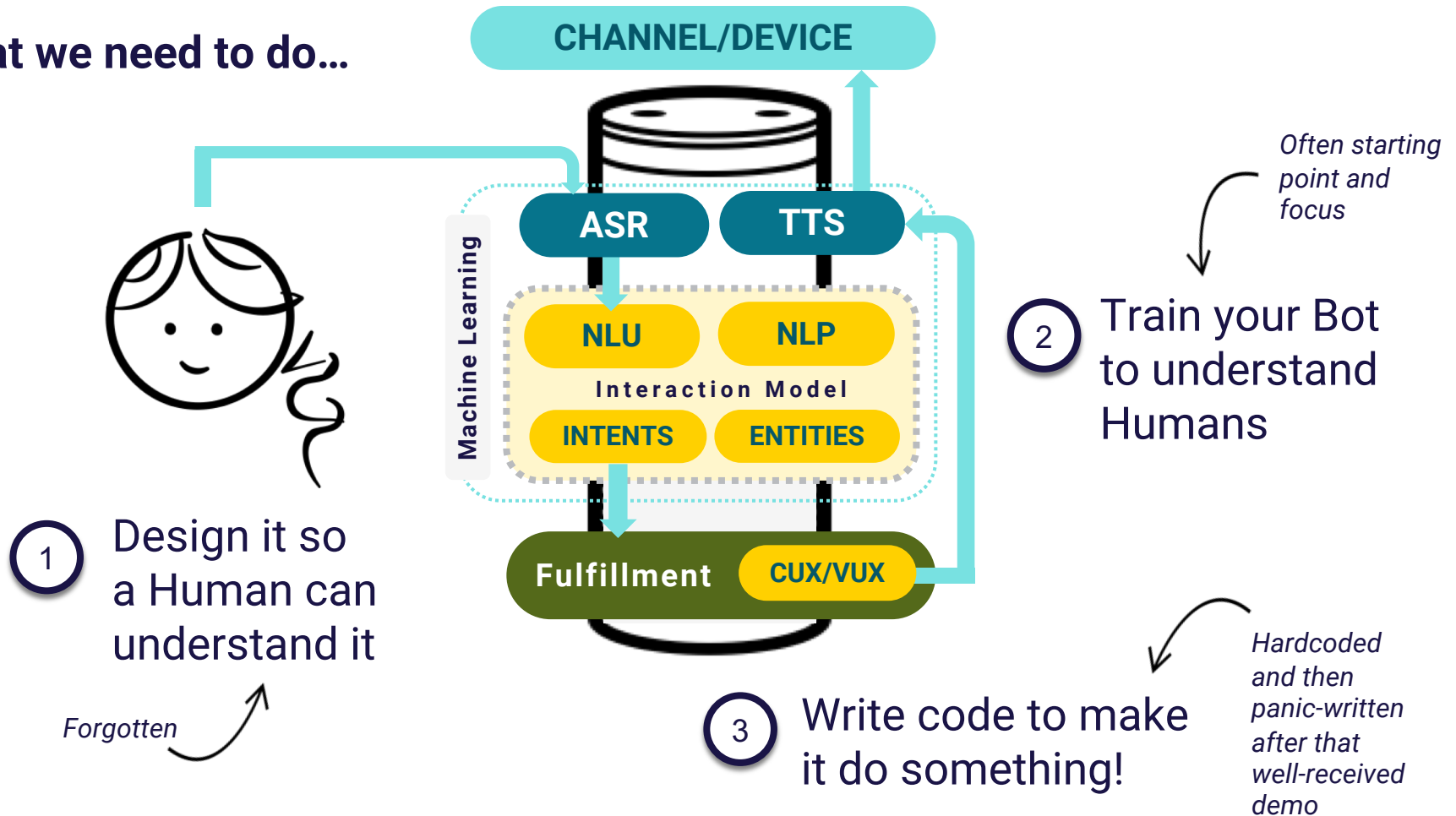




What we need to do...

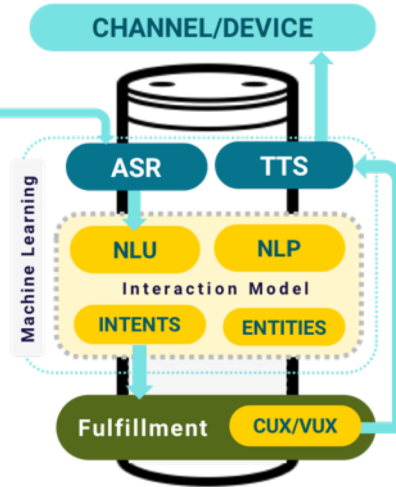


What we need to do...



1

Design it so a Human can understand it



2 Train your Bot to understand Humans

3 Write code to make it do something!



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

BEST VIEWED WITH INTERNET EXPLORER Ver 6+ WHY?
TO HEAR AWESOME MUSIC DO NOT BLOCK POPUNDER
Ads Rates Site Stats WeBLOG-4/19/13
LINKS/View/Add FREE Classifieds/View/Add
Welcome to the hosanna1.com Portal Page
Home of **AAA World Wide Web Design**
& Hosanna Afghan Hounds

USE REFRESH BUTTON - We Update OFTEN FULL SCREEN SETTING BEST
ARCHIVES | **AFGHAN HOUNDS:** Blue Sky | Gwich'in |
Hosanna+PUPPIES! | Mahadi
travel-hound | Other DOG Breeds: **Cyrano Hounds** (whippets/greyhounds)
| **Play Kenne!** (KEES/PYRS) KEESHOND PUPPIES HERE NOW!
SKY Shelties | **ART/CobraOriginals** Afghan, dog breeds, horses
HORSES: **Step Of Faith Farm** | SPIRITUAL: 911 | AALF

NEVER FORGET!

Welcome to
hosanna1.com
HOME OF
Afghan Hounds
Design



Design Matters!

Not a CSS and HTML problem!



I WILL DWELL IN THE HOUSE
Hosanna Afghans HISTORY Page [CLICK HERE](#)

To learn about
Christians who smoke,
and smoke in the Bible,
[CLICK HERE](#)

these 'most beautiful dogs in the world, please check out THIS PAGE
COME SEE THE MOST BEAUTIFUL ART CAR IN THE WORLD, AWARD WINNING PARADE CAR, AND OUR HOSANNA AFGHAN HOUNDS DOG SHOW

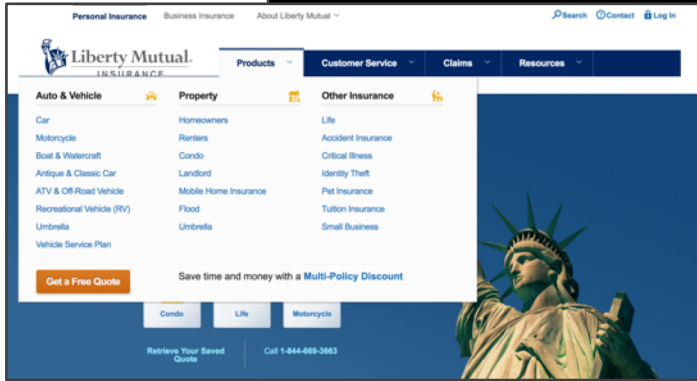
VISIT **AIF** Samson Happy Precious Ruby Bliss Red Man

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

```
Starting MS-DOS...

HIMEM is testing extended memory...done.

C:\>C:\DOS\SMARTDRV.EXE /X
C:\>_
```



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

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

The screenshot shows a web form for Liberty Mutual Insurance. The header includes the Liberty Mutual logo and navigation links for 'Auto', 'Quote', and 'Purchase Online'. The main heading is 'Your Info'. Below this, there are three green callout boxes: 'You're just a few steps away from your free quote.', 'It's simple. Get your quote in minutes.', and 'It's safe. Your information will never be sold.' The 'Address' section includes a sub-heading 'If you are moving, enter your future address.' and fields for 'Street Address', 'Apt/Unit #', 'ZIP Code' (with '03801' entered), and 'City' (with 'Portsmouth' entered and a dropdown for 'NH'). There are two radio button questions: 'Is this your mailing address?' (with 'Yes' selected) and 'Do you currently live here?' (with 'No' selected). A 'Save & Continue' button is at the bottom right.

The screenshot shows a paper W-9 form titled 'Request for Taxpayer Identification Number and Certification'. The form includes sections for 'Name', 'Address', 'Employer Identification Number (EIN)', and 'Certification'. The 'Certification' section contains several checkboxes and text boxes for the taxpayer to complete, including a section for 'Signature' and 'Date'.


```
Starting MS-DOS...

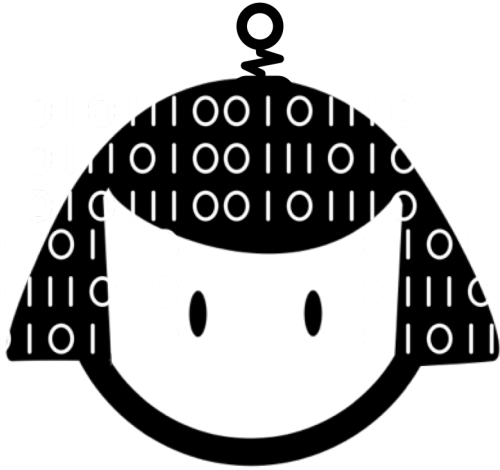
HIMEM is testing extended memory ..done.

C:\>C:\DOS\SMARTDRV.EXE /X
C:\>_
```

The screenshot shows the Liberty Mutual Insurance website. The navigation bar includes 'Personal Insurance', 'Business Insurance', and 'About Liberty Mutual'. A search bar and 'Log In' link are also present. The main content area is divided into three columns: 'Auto & Vehicle' (listing Car, Motorcycle, Boat & Watercraft, etc.), 'Property' (listing Homeowners, Renters, Condo, etc.), and 'Other Insurance' (listing Life, Accident Insurance, etc.). A 'Get a Free Quote' button is visible at the bottom left. The background features a large image of the Statue of Liberty.

The screenshot shows the 'Your Info' section of the Liberty Mutual website. It includes a progress indicator 'You're just a few steps away from your quote.' and three status items: 'It's simple. Get your quote in minutes.', 'It's safe. Your information will never be sold.', and 'It's secure. Your information is protected with 256-bit encryption.' Below this are input fields for 'Address', 'Age/Unit #', 'ZIP Code' (with '03801' entered), and 'City' (with 'Portsmouth' and 'NH' entered). There are also radio buttons for 'Is this your mailing address?' (Yes selected) and 'Do you currently live here?' (Yes selected). A 'Save & Continue' button is at the bottom right.

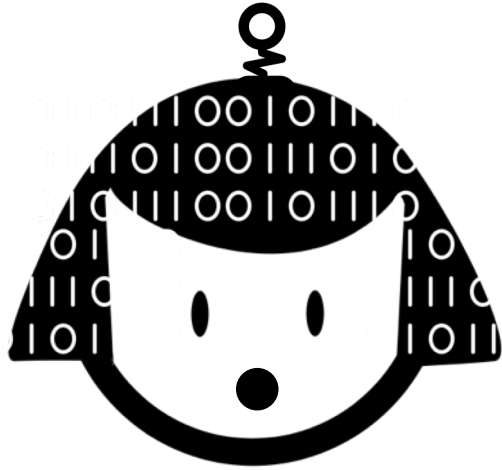
The screenshot shows a W-9 form titled 'Request for Taxpayer Identification Number and Certification'. It includes fields for 'Name', 'Address', 'SSN', and 'Mailing Address'. There are checkboxes for 'Individual', 'Partner', 'Sole proprietor or disregarded entity', 'Limited liability company (LLC) or partnership', 'Trust or estate', and 'Nonresident alien'. A 'Print' button is at the bottom right.



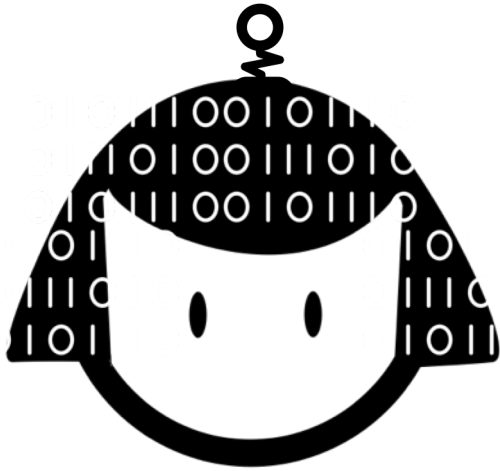
**People are the original
conversational interface**

Model your design on a
human, not a website



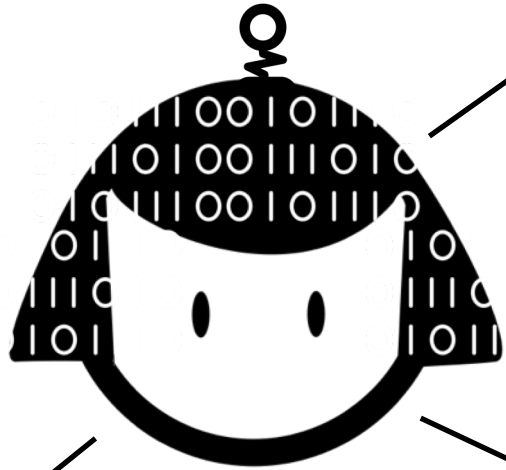


When you start with
thinking about trying out
a design you may
discover that a chatbot
isn't even the right thing
to build...



But assuming it is...

*1 minute
Conversational
Design
overview*



Be Clear

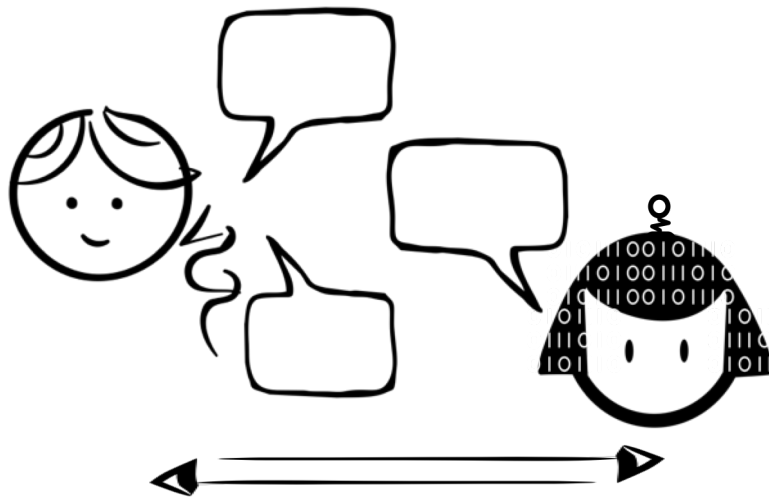
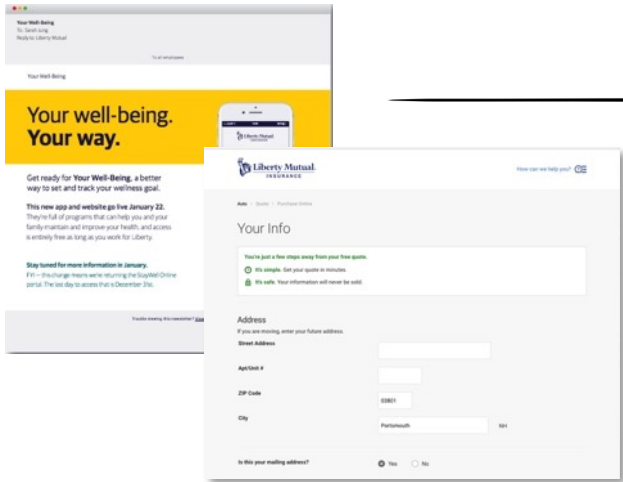
- Keep your personality consistent
- Keep answers short and to the point
- Set expectations – what can you do
- Reflect back key information and confirm before critical actions.

Be Helpful

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

Be Nice

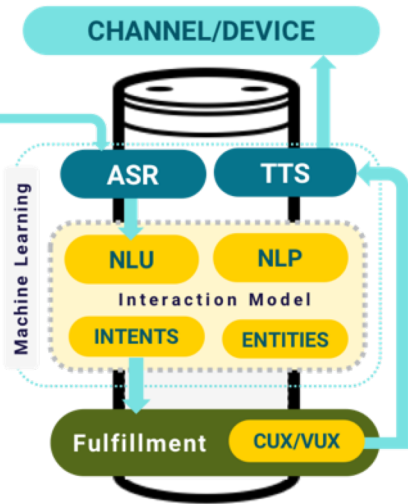
- Take the blame – in a conversation the human is never wrong
- Chatbots aren't smarter than a human
- Be understanding if people don't want to talk to you – offer alternatives.



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

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

1 Design it so a Human can understand it



3 Write code to make it do something!

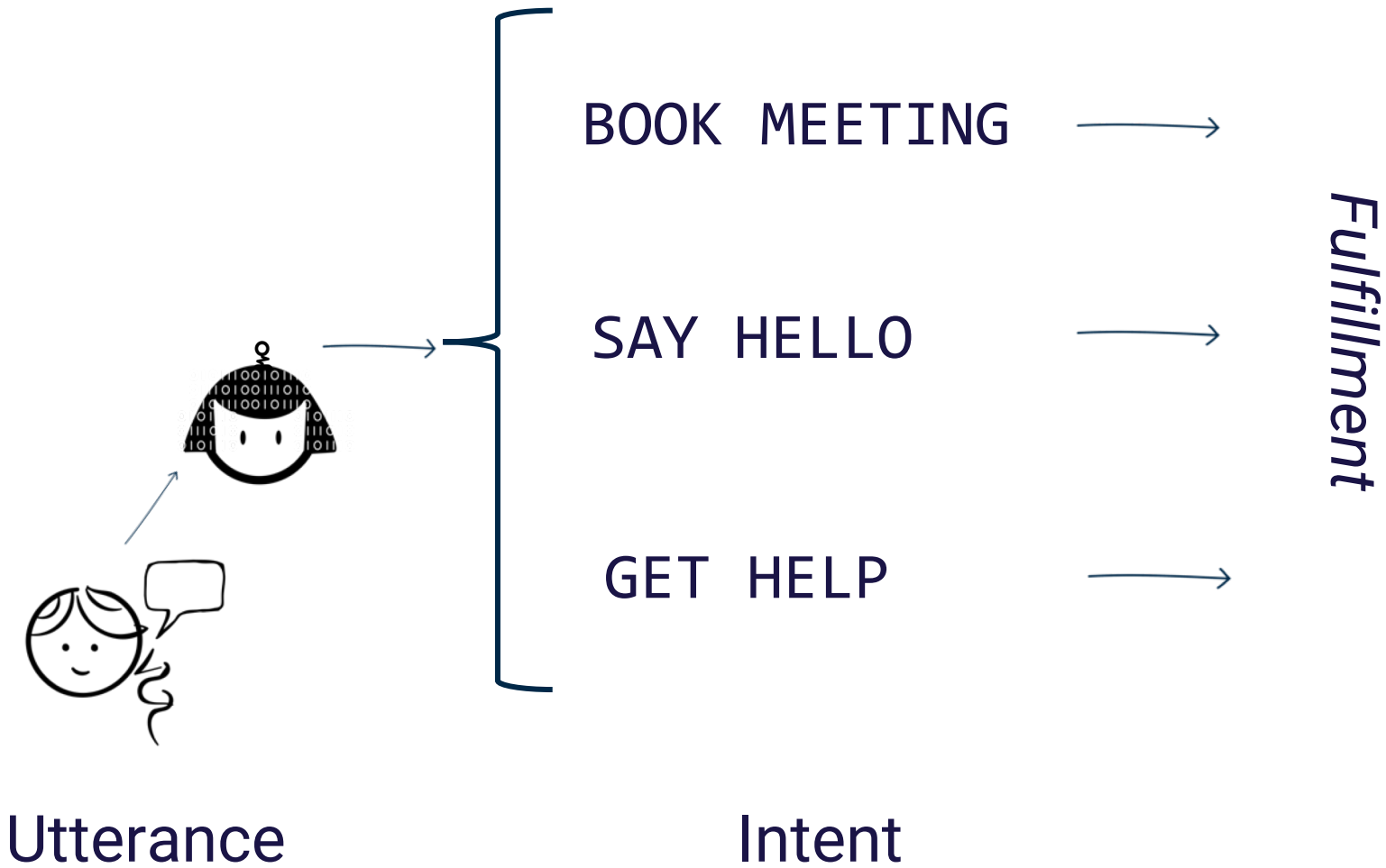
2 Train your Bot to understand Humans

Amazon Lex Console

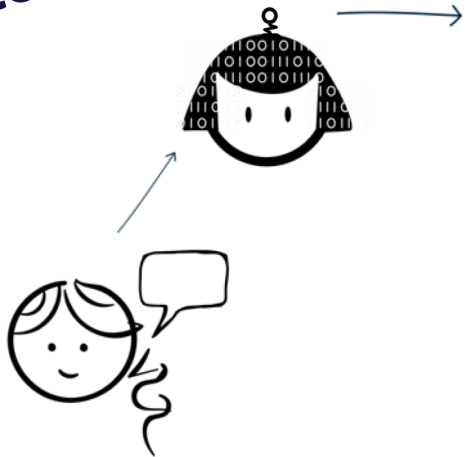
The screenshot displays the Amazon Lex console interface for a chatbot named "DemoAssistantBot". The "Intents" section is expanded to show the configuration for "demo_BookMeetingIntent". Under "Sample utterances", there are 15 entries, each with a text input field and highlighted slot values in colored boxes (purple for time, green for person, orange for date). The utterances include phrases like "please book a meeting at [time]", "i want to book a meeting", "i need a meeting booked", "can you book a meeting with [person] at [time] on [date]", "book me a meeting [time]", "book me a meeting with [person]", "book me 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]", and "book a meeting with [person]".

On the right side, the "Test bot (Latest)" panel is active, showing a green status "Ready, Build complete." and a text area with the message: "You're now ready for testing. Type an utterance below to begin conversation with your chatbot." Below this is an "Inspect response" section with a "Hide" button.

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



As simple
providing a set
of **sample**
utterances!



Utterance

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
...

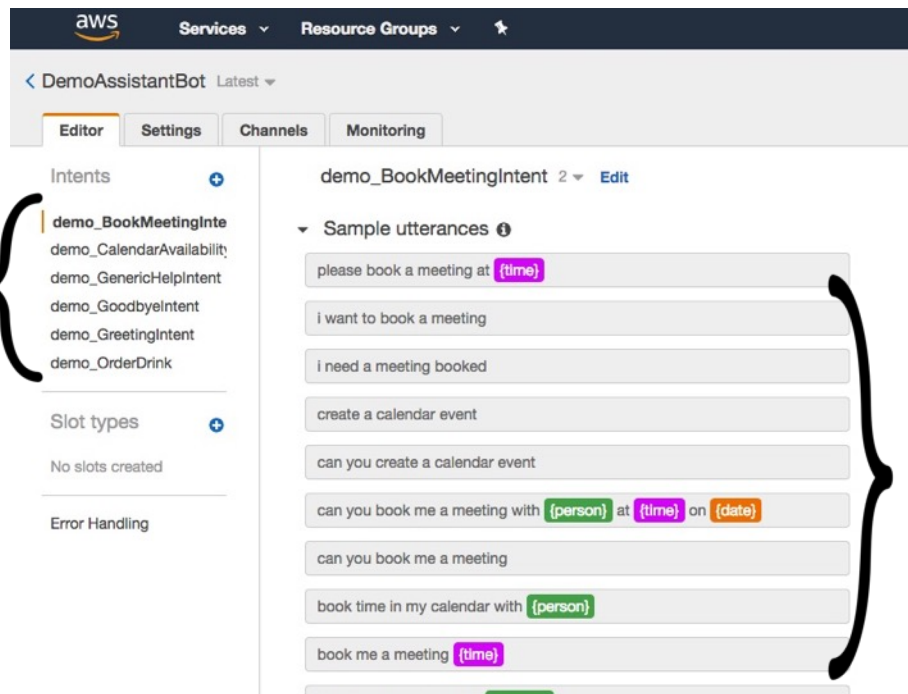
GET HELP

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

Intent

Amazon Lex Console

SET OF INTENTS
Represents the functionality the bot has



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

Amazon Lex Console

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

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

▼ Lambda initialization and validation ⓘ

Initialization and validation code hook

▼ Slots ⓘ

Priority	Required	Name	Slot type	Version
		e.g. Location	e.g. AMAZON.US_CITY	
1.	<input checked="" type="checkbox"/>	numberOfAttendees	AMAZON.NUMBER	Built-in ▼
2.	<input checked="" type="checkbox"/>	person	AMAZON.Person	Built-in ▼
3.	<input checked="" type="checkbox"/>	date	AMAZON.DATE	Built-in ▼
4.	<input checked="" type="checkbox"/>	time	AMAZON.TIME	Built-in ▼

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

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

Amazon Lex Console

▼ Slots ⓘ

Priority	Required	Name	Slot type
		e.g. Location	e.g. AMAZON.US_CITY
1. ▼	<input checked="" type="checkbox"/>	numberOfAttendees	AMAZON.NUMBER
2. ^ ▼	<input checked="" type="checkbox"/>	person	AMAZON.Person
3. ^ ▼	<input checked="" type="checkbox"/>	date	AMAZON.DATE
4. ^	<input checked="" type="checkbox"/>	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 ▼

FULFILLMENT location defined.

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

Amazon Lex Console

Build to create the Interaction Model

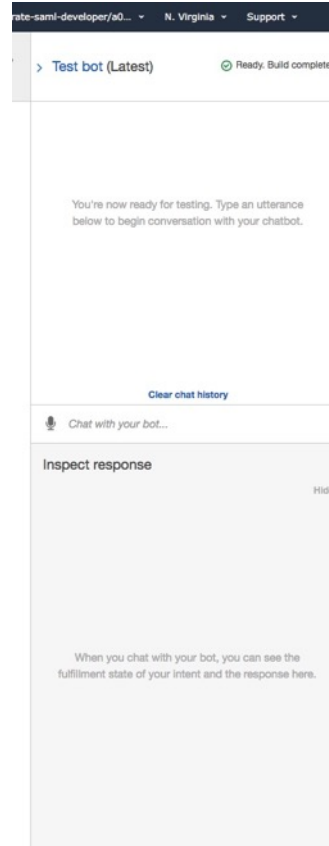
Publish to make the functionality live to users.

The screenshot displays the Amazon Lex console interface for a bot named 'DemoAssistantBot'. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information. The main content area is divided into several sections:

- Intents:** A list of intents including 'demo_BookMeetingIntent', 'demo_CalendarAvailabilityIntent', 'demo_GenericHelpIntent', 'demo_GoodbyeIntent', 'demo_GreetingIntent', and 'demo_OrderDrink'.
- Slot types:** A section indicating 'No slots created'.
- Error Handling:** A section for configuring error handling.
- demo_BookMeetingIntent:** A detailed view of this intent, showing a list of 'Sample utterances' such as '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]', and 'book me a meeting with [person]'.
- Build and Publish buttons:** Two buttons are visible at the top right of the console. An arrow points from the text 'Build to create the Interaction Model' to the 'Build' button. Another arrow points from the text 'Publish to make the functionality live to users.' to the 'Publish' button.
- Test bot (Latest):** A panel on the right side of the console, currently showing a message: 'You're now ready for testing. Type an utterance below to begin conversation with your chatbot.' Below this is a 'Clear chat history' link and a chat input field with the placeholder 'Chat with your bot...'. An 'Inspect response' button is also visible.

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 displays the Amazon Lex console interface for configuring a channel. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information. The main header shows the bot name 'DemoAssistantBot' and the 'Channels' tab is selected. A sidebar on the left lists available channels: Facebook, Kik, Slack, and Twilio SMS. The main content area is titled 'Facebook' and contains a form with the following fields:

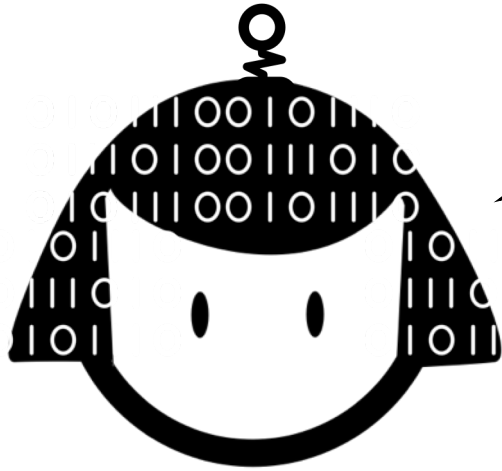
- Channel Name* (text input)
- Channel Description (text input)
- IAM Role: AWSServiceRoleForLexChannels (pre-filled, with a note 'Automatically created on your behalf')
- KMS Key: aws/lex (dropdown menu)
- Alias* (dropdown menu)
- Verify Token* (text input, placeholder: Verify Token)
- Page Access Token* (text input, placeholder: Page Access Token)
- App Secret Key* (text input, placeholder: App Secret Key)

An 'Activate' button is located at the bottom of the form. A footer bar contains 'Feedback', 'English (US)', and copyright information.

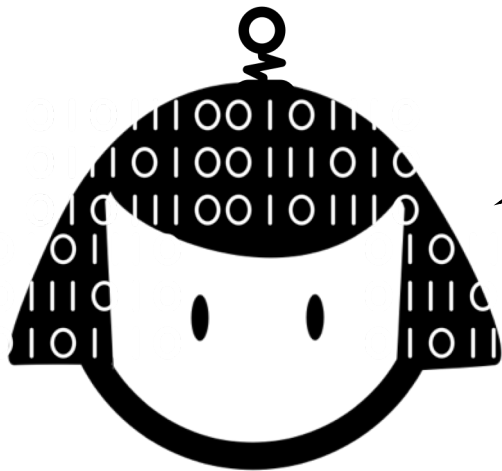
Amazon Lex Console

The screenshot displays the Amazon Lex console for a bot named 'DemoAssistantBot'. The current view is the 'Editor' tab for the 'demo_BookMeetingIntent'. The left sidebar lists several other intents: demo_CalendarAvailability, demo_GenericHelpIntent, demo_GoodbyeIntent, demo_GreetingIntent, and demo_OrderDrink. Below the intent list, it shows 'Slot types' (No slots created) and 'Error Handling'. The main area shows 'Sample utterances' for the selected intent, with each utterance and its fulfillment state (e.g., (time), (person), (date)) displayed in a text box. The right sidebar contains instructions for testing the bot, including a 'Test bot (Latest)' button and a 'Ready, Build complete.' status. A large blue starburst graphic with a wand is overlaid on the right side of the console, pointing towards the text 'Chatbot Done...?'.

Chatbot Done...?



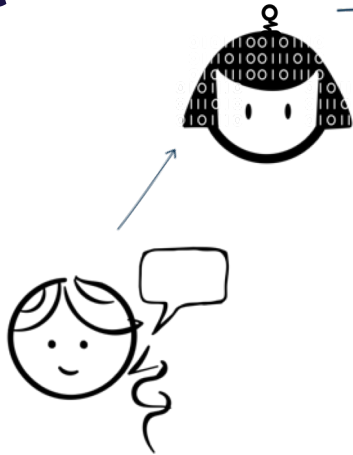
Sorry, I'm just a
baby bot. I'm still
learning.



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

are you?

As simple
providing a set
of **sample**
utterances!

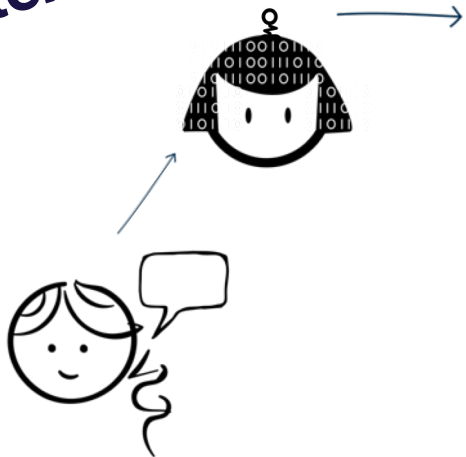


Utterance

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 ...

Intent

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

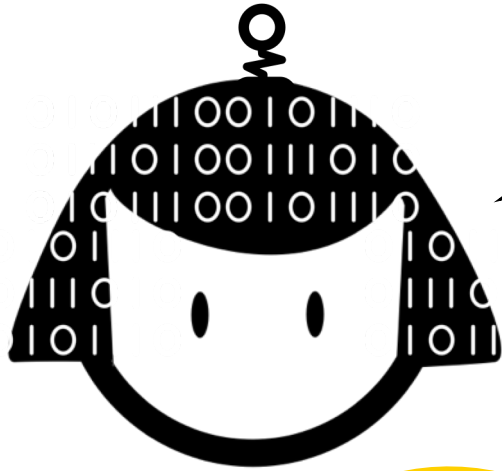


Utterance

Model your design on a
human

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

Intent



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

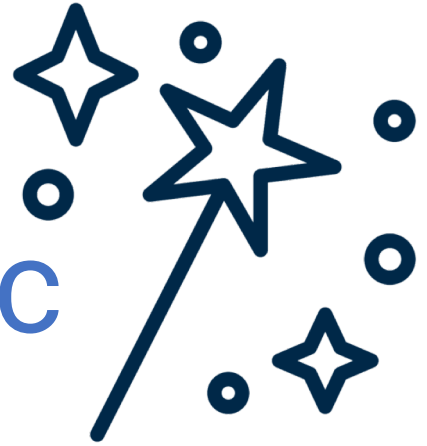
Look for **Interaction Patterns**

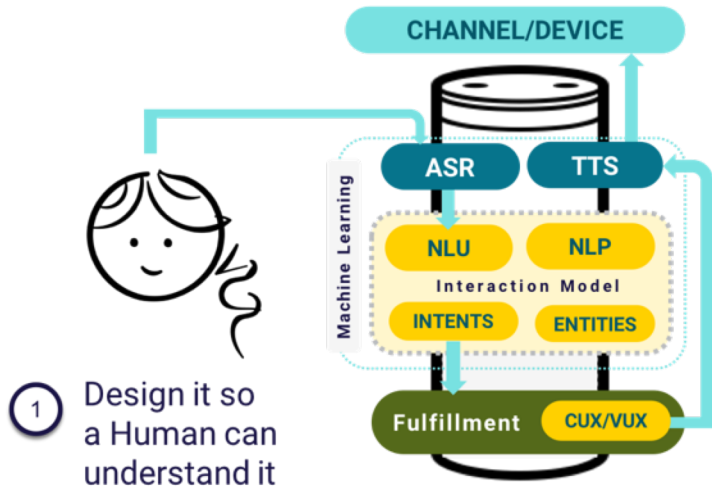
Can we improve the conversation flow

Learn
Fast

Change
Fast

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

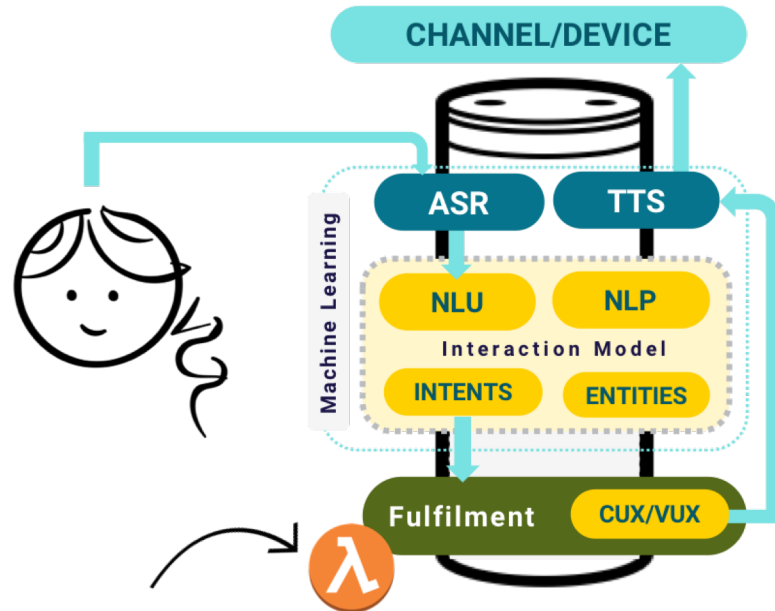




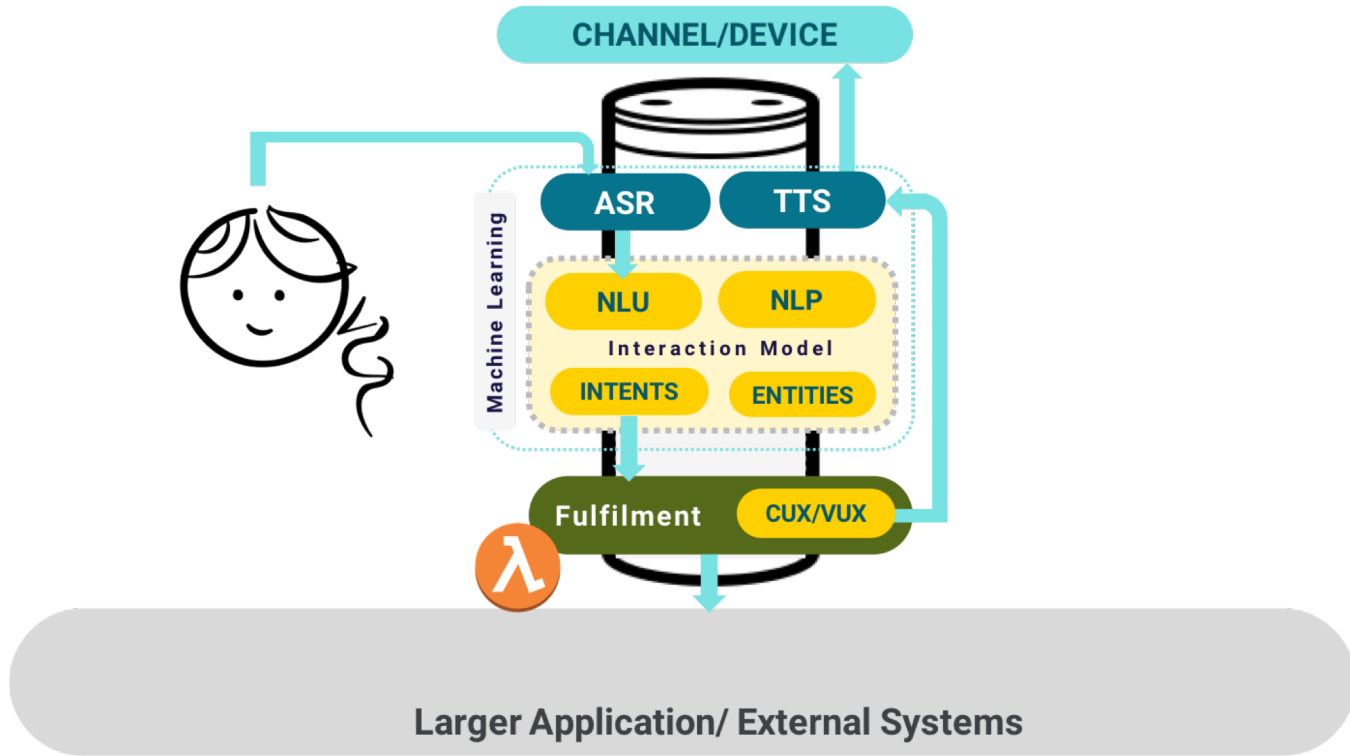
1 Design it so a Human can understand it

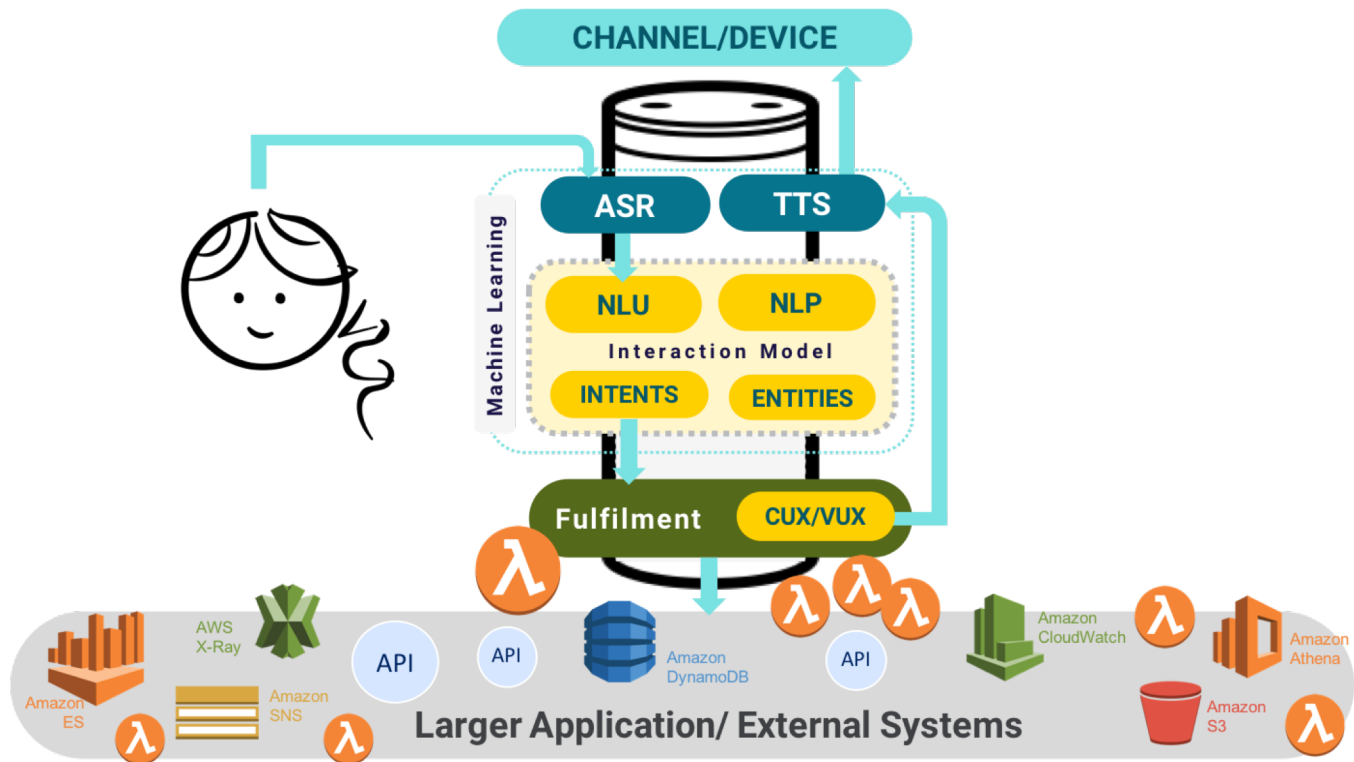
2 Train your Bot to understand Humans

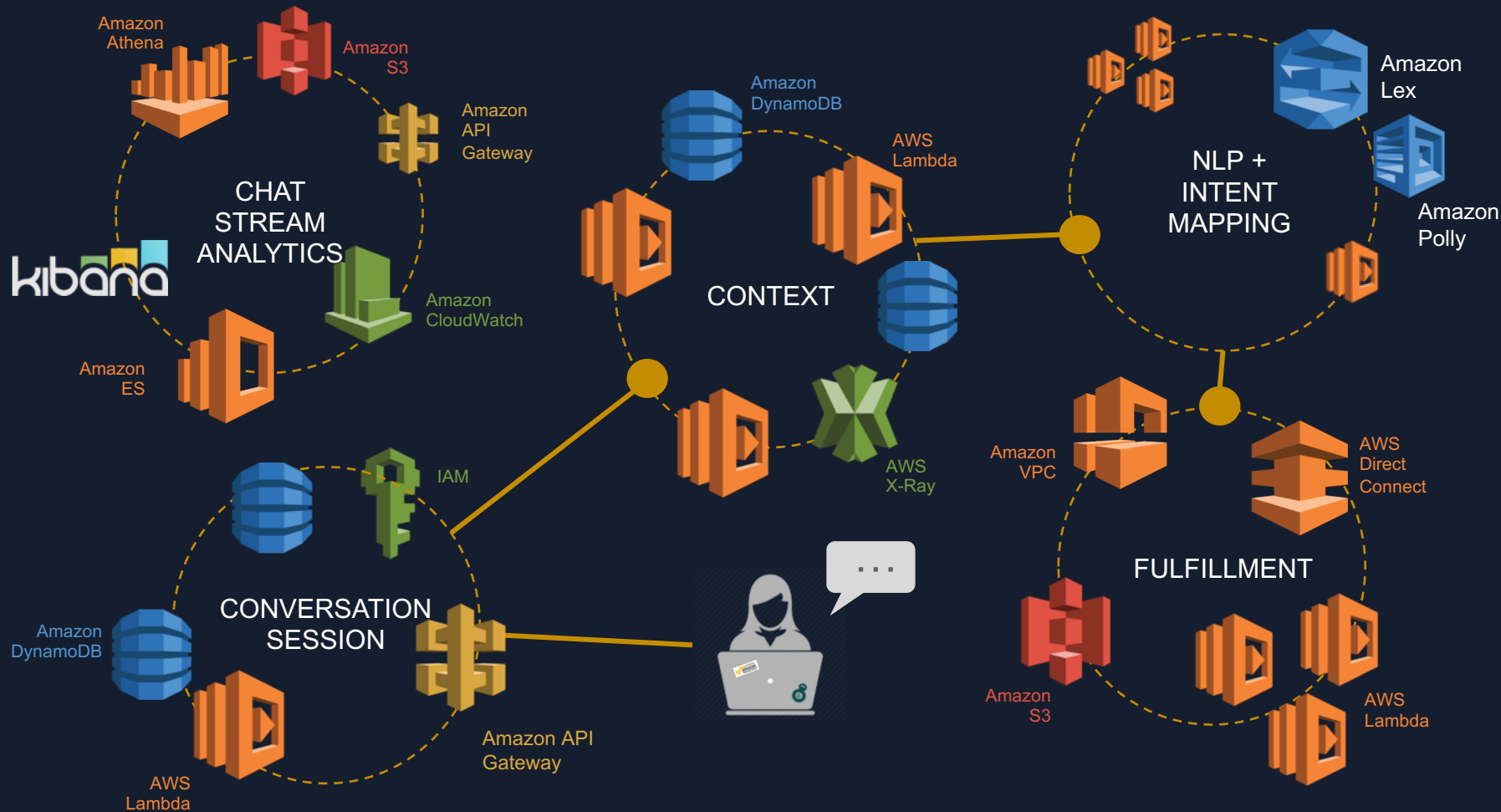
3 Write code to make it do something!



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







THE DIGITAL ASSISTANT BOT PLATFORM ARCHITECTURE



229

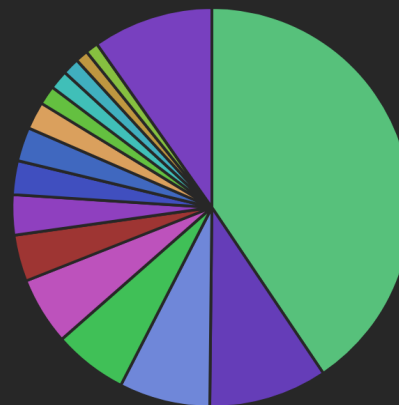
Number of Unique Sessions

166

Unique Users

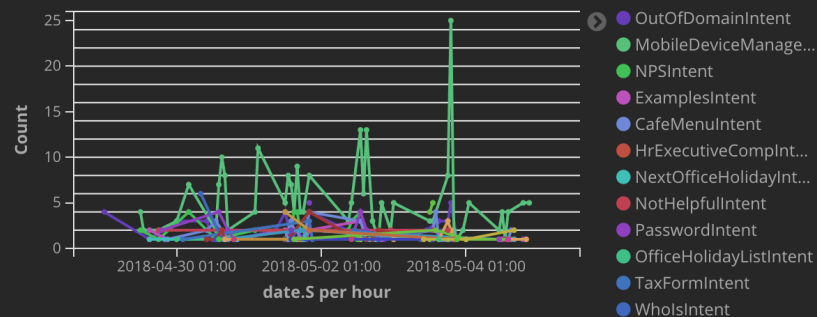
Intent Hits Table

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



- MobileDeviceManage...
- OutOfDomainIntent
- CafeMenuIntent
- NPSIntent
- ExamplesIntent
- QuickLinkIntent
- PasswordIntent
- HelpDeskIntent
- WholsIntent
- ThanksIntent
- BrowserTroublesho...
- NextOfficeHolidayInt...
- HrServiceCenterGene...
- DefinitionService
- GreetingIntent
- Other

Intent hits (short timeline)

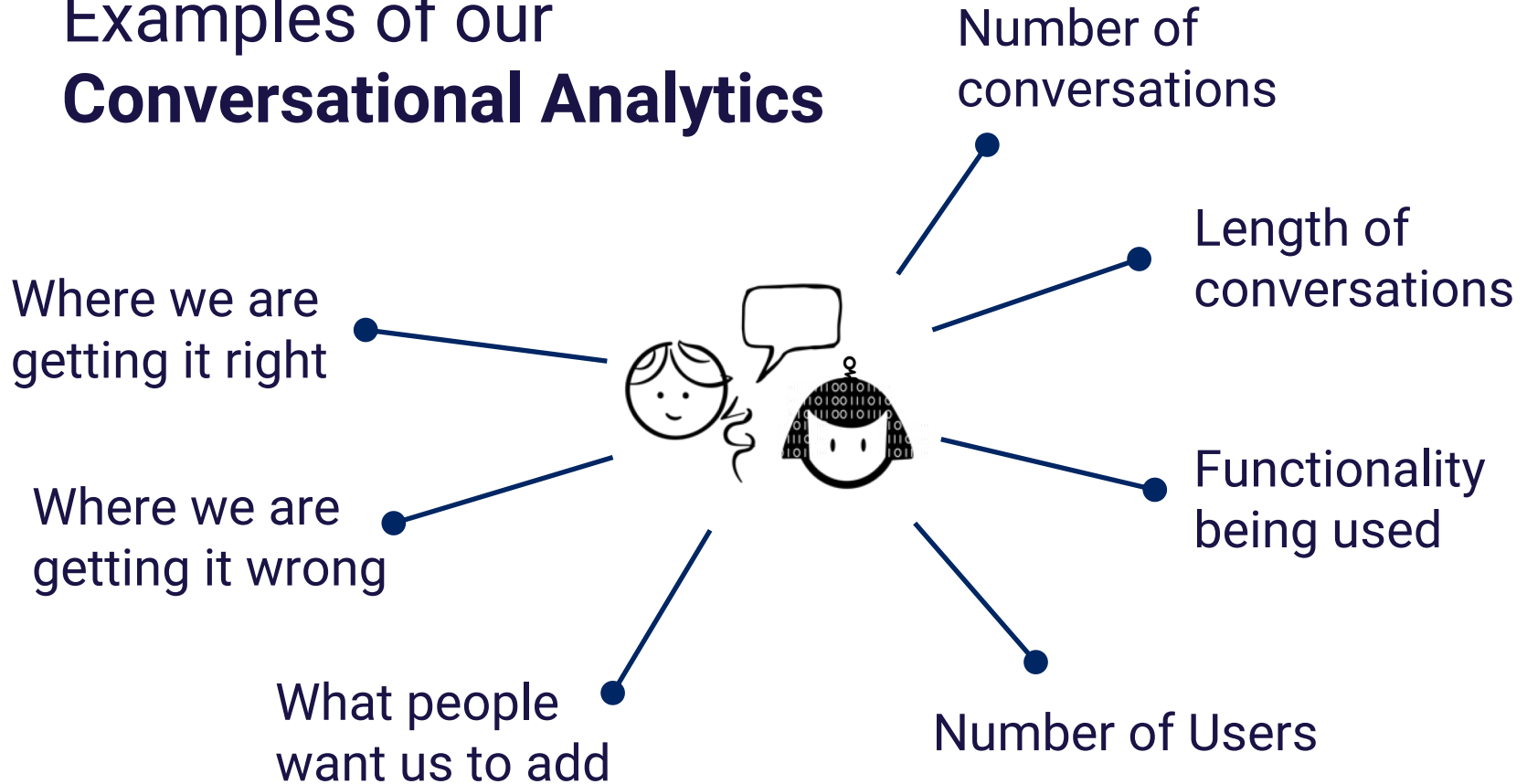


Hosting - Completed Conversations By A...

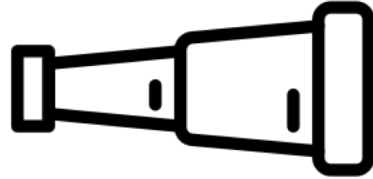
Action

Count

Examples of our **Conversational Analytics**



Observability is critical



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

Why Serverless??

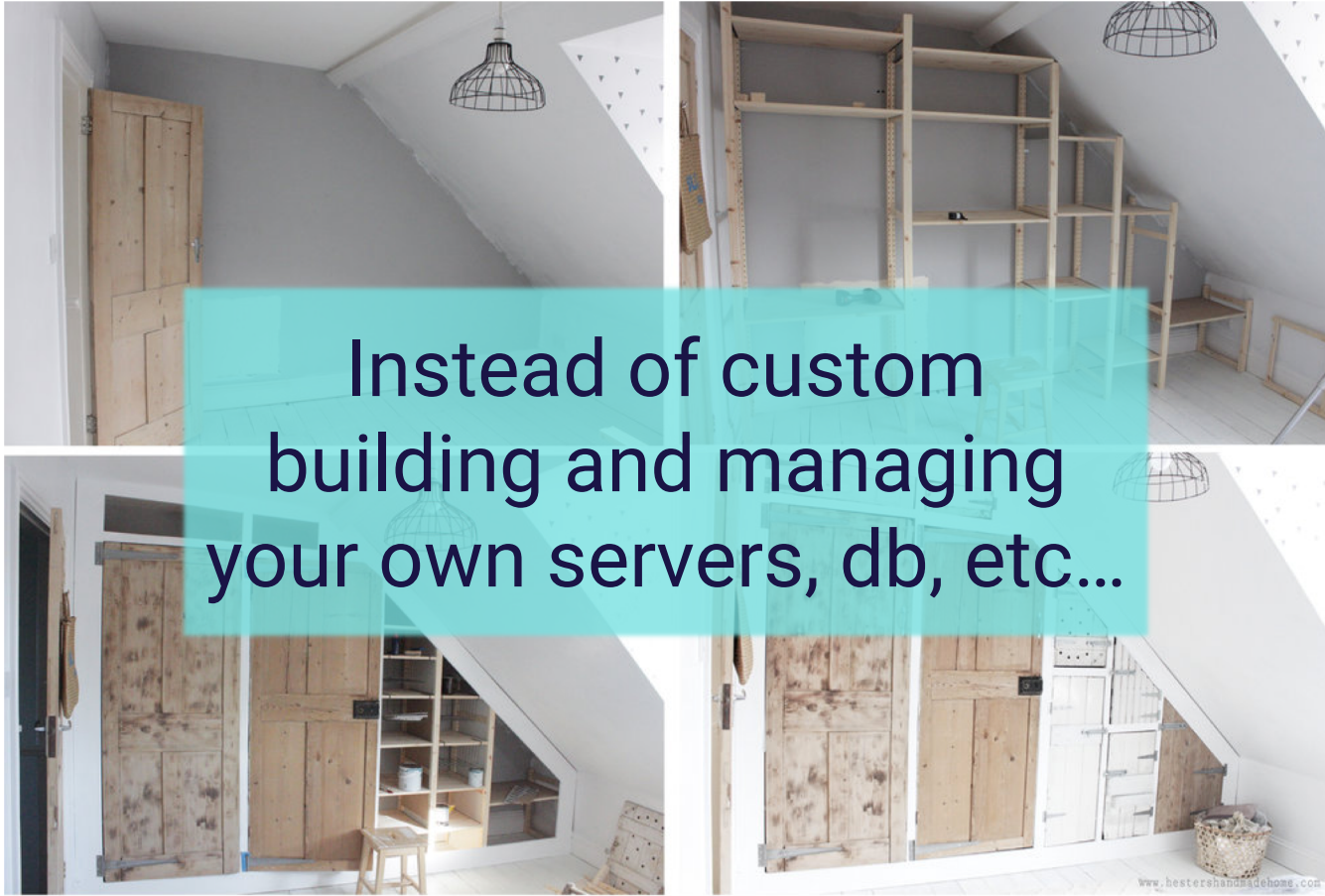


Lower Cost
lets us



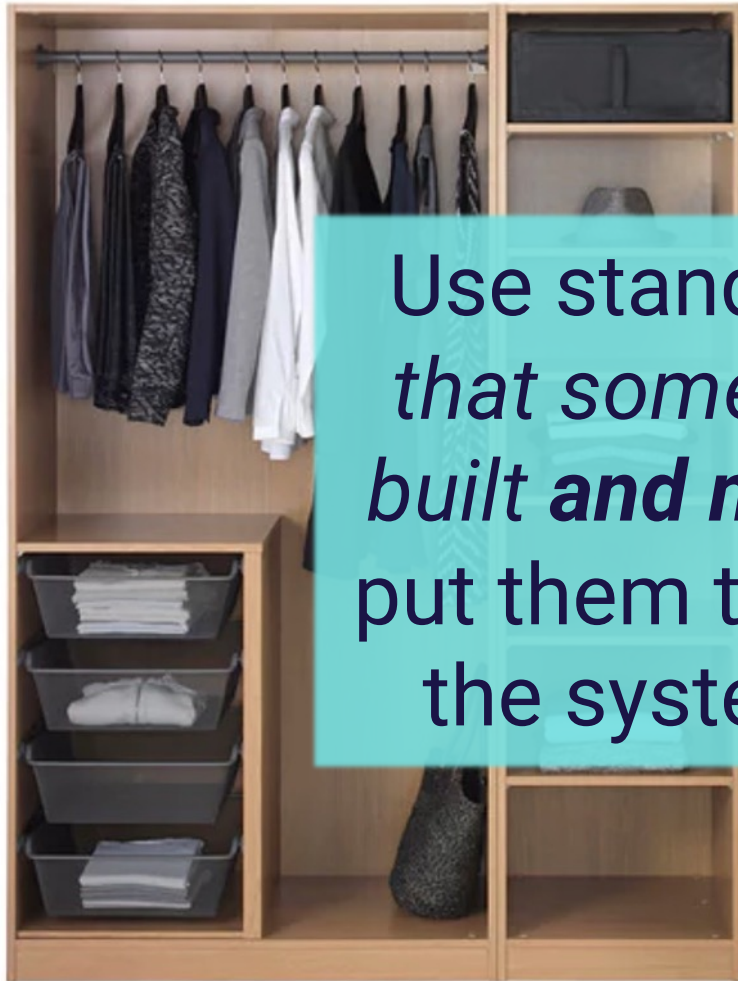
Off-loading Maintenance
lets us



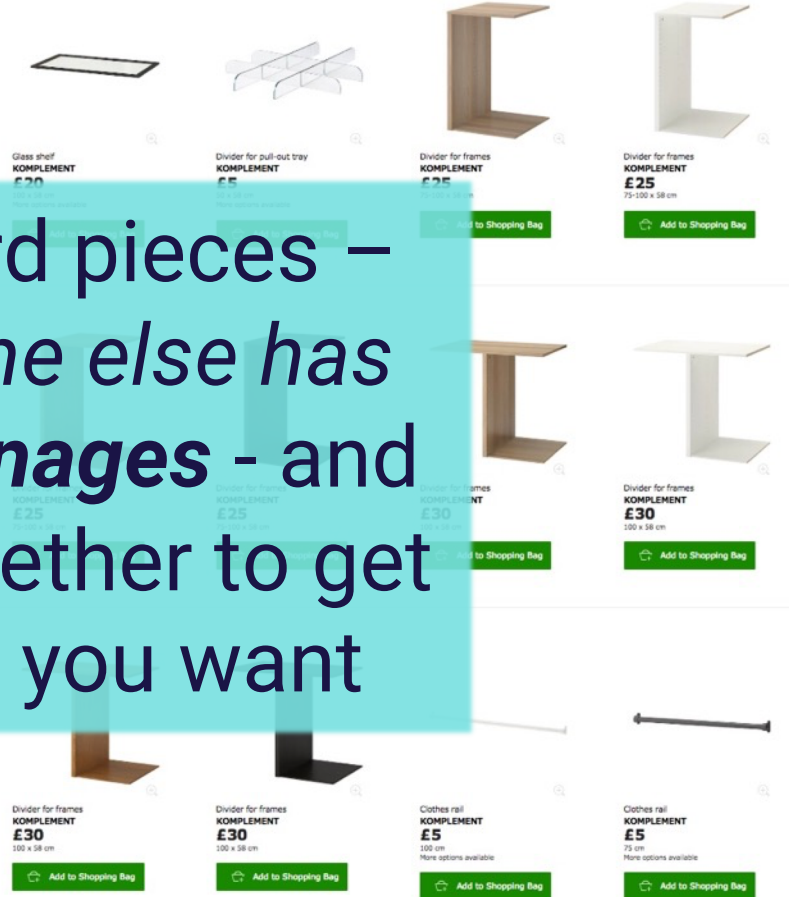


Instead of custom
building and managing
your own servers, db, etc...

wardrobe by www.hestershandmadehome.com



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

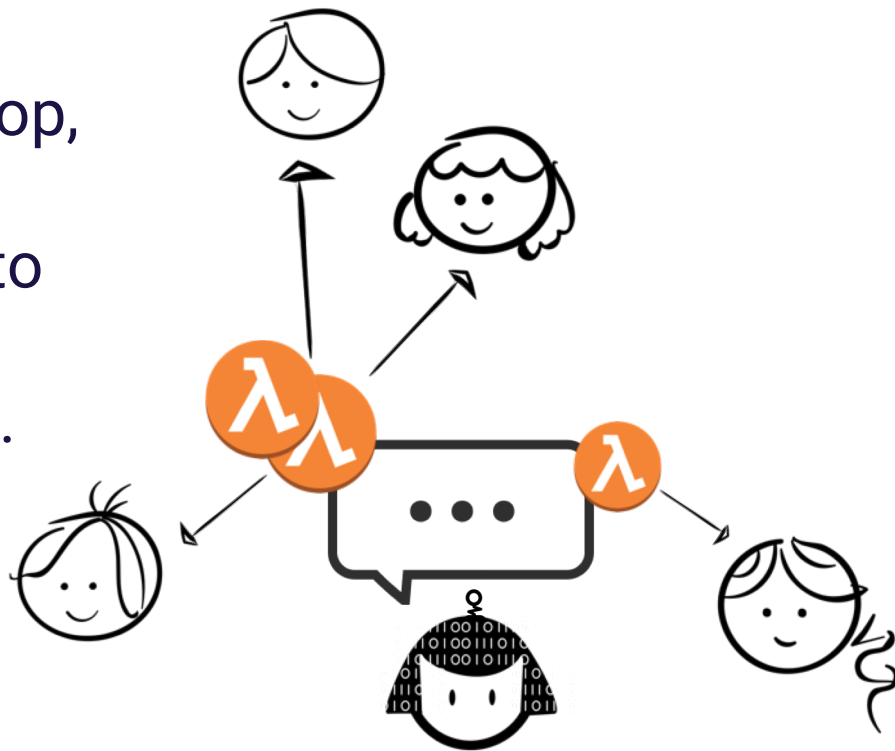


Serverless

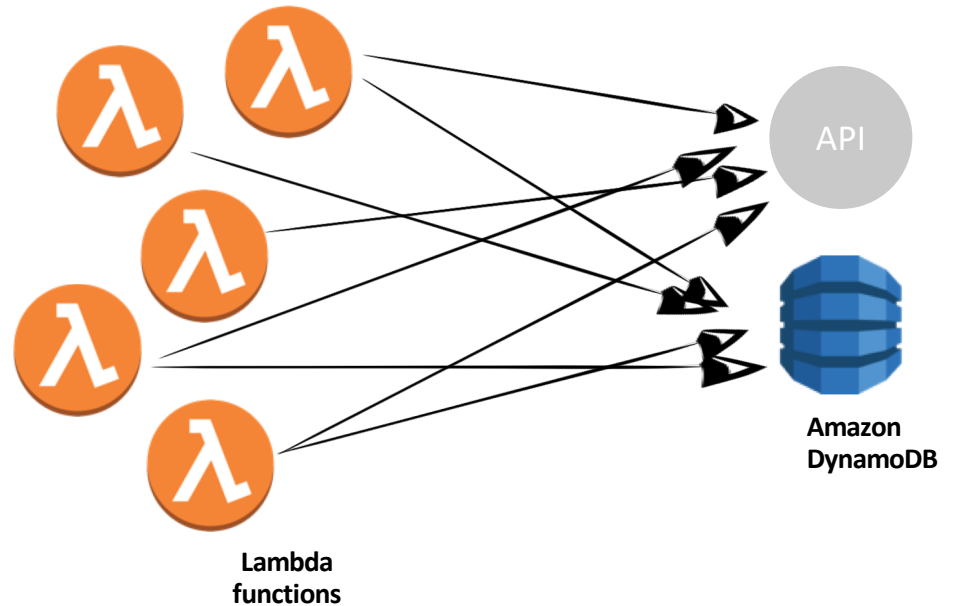


Services

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

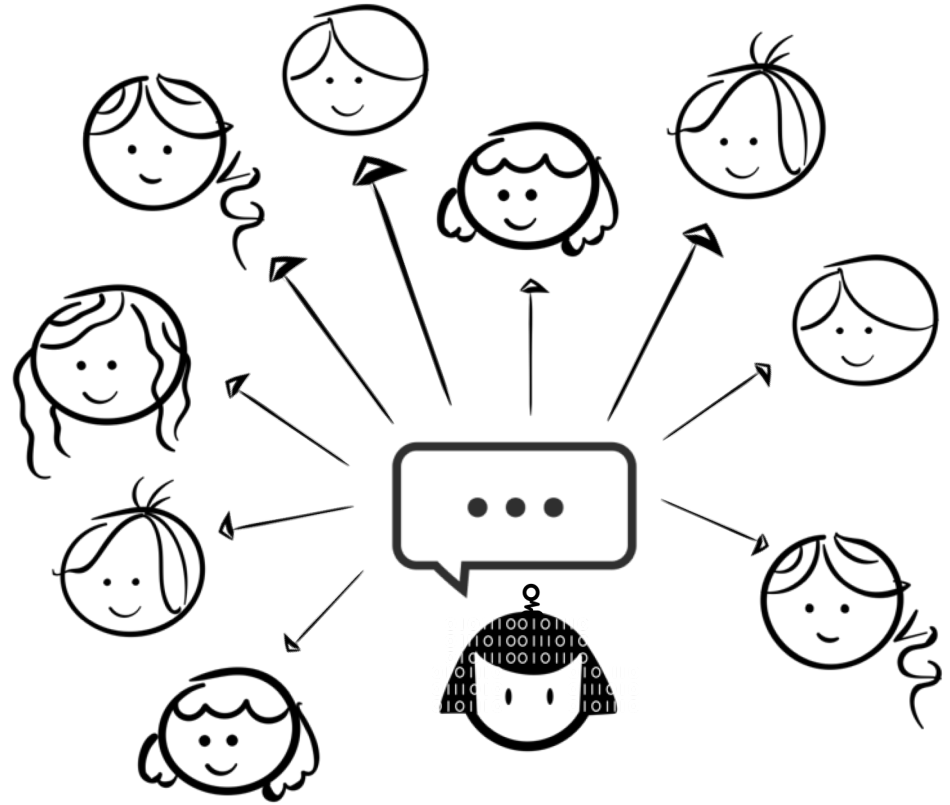
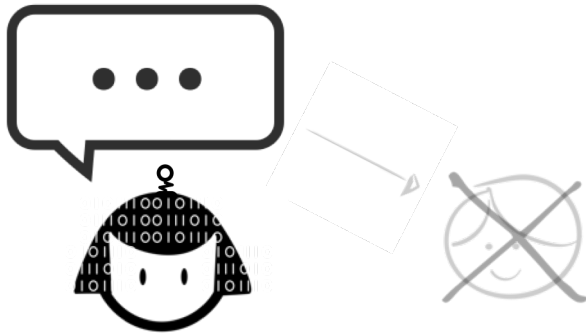


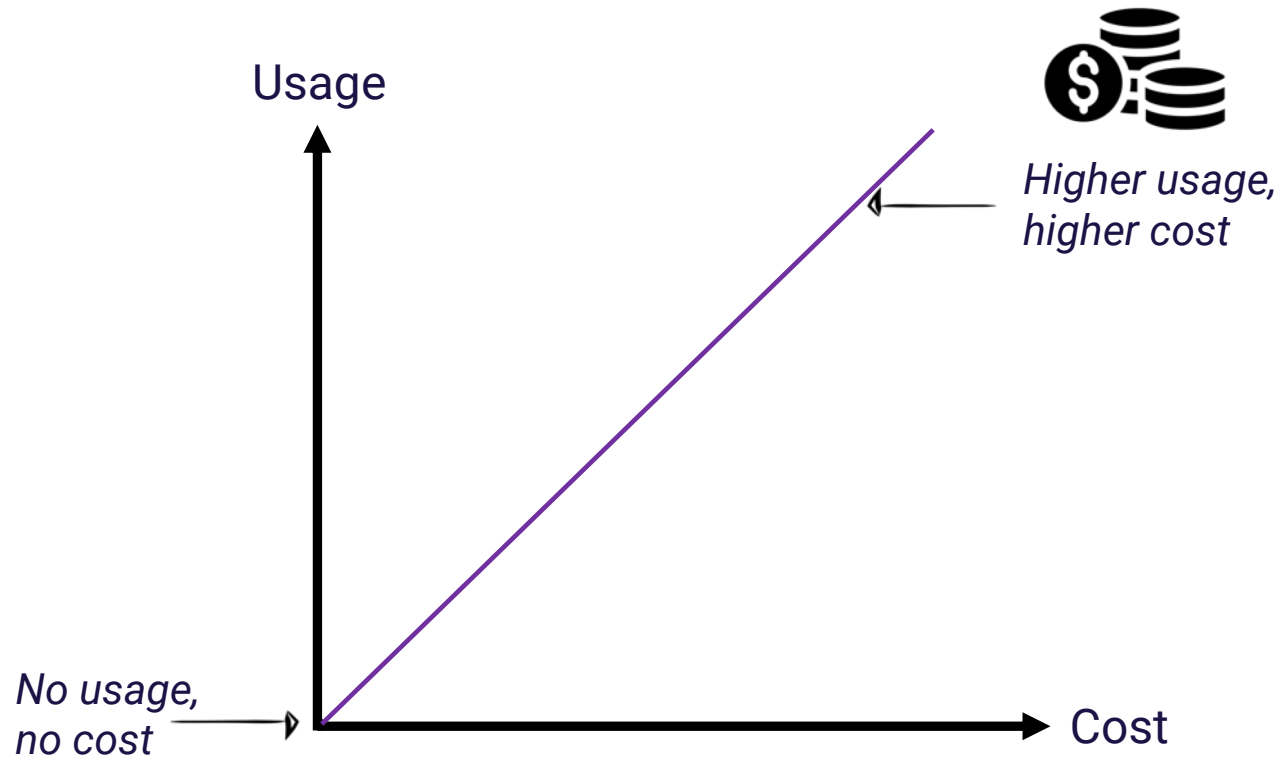
Ensure that *everything* **scales** - or at least know your **limits**.

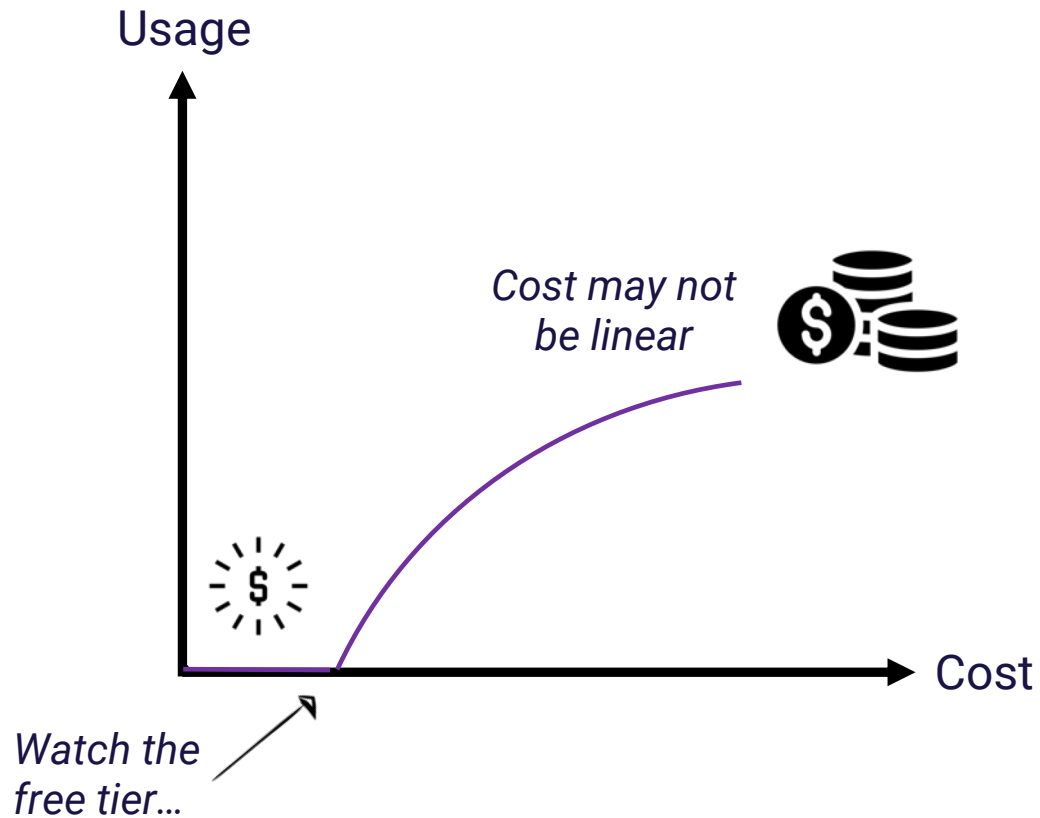


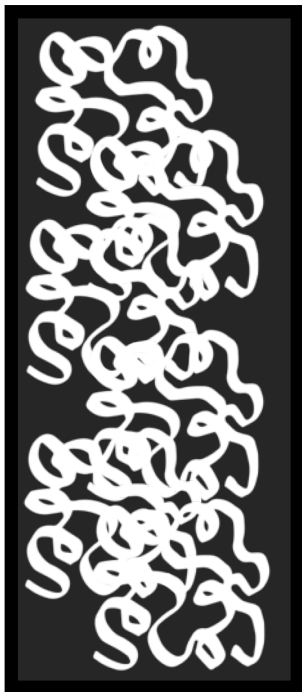
Serverless means you don't pay if **no-one** talks to your bot...

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



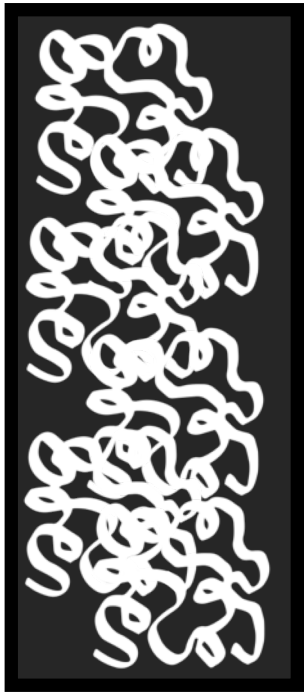




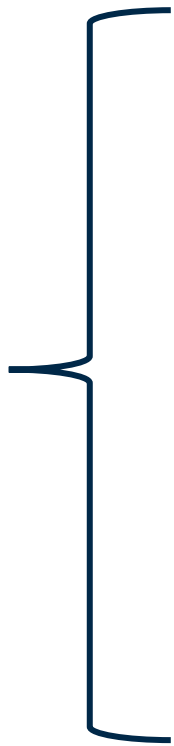
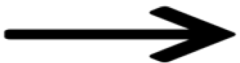


Monolith

Some notes for
those with
existing systems



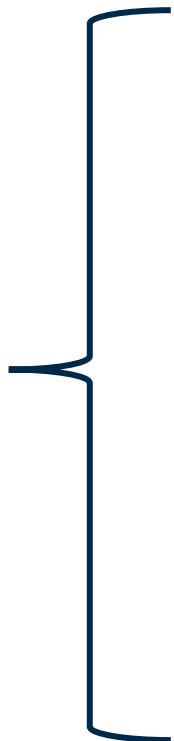
Monolith



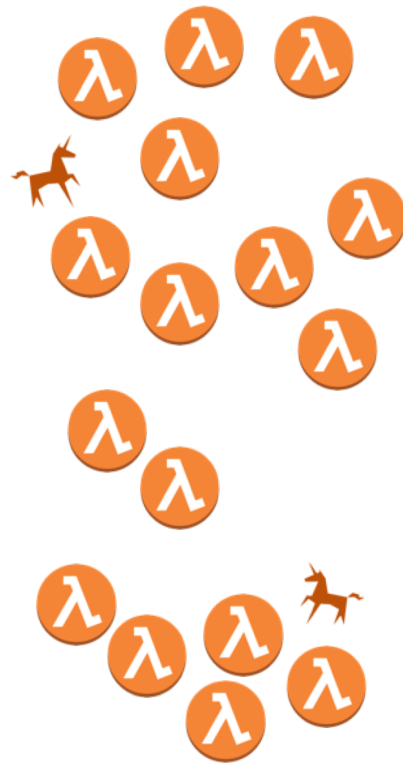
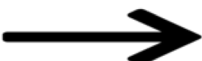
Microservices



Monolith

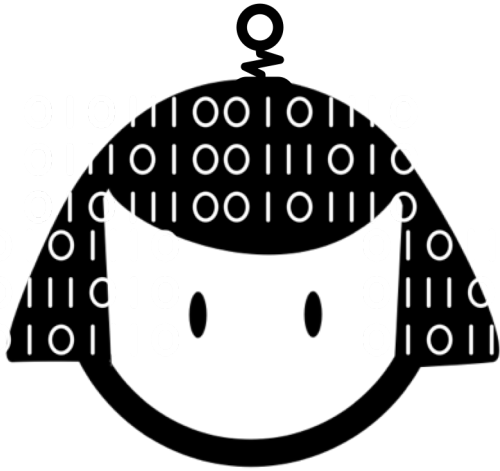


Microservices



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

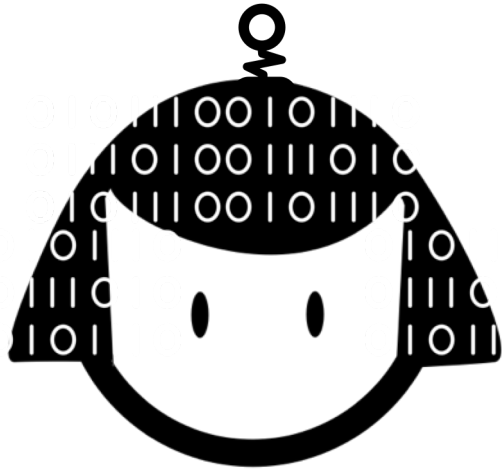


A good human conversation is
synchronous

You **wait** for a response

You **expect** a response
immediately

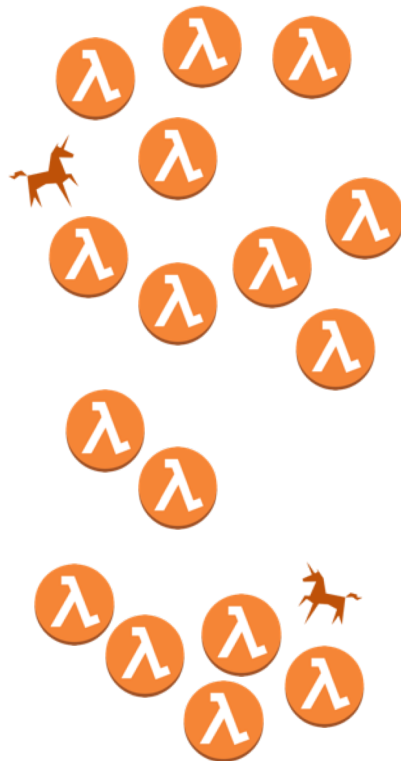




A Conversational Architecture needs to

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

This is particularly critical for a voice-based bot



Monolith



Microservices

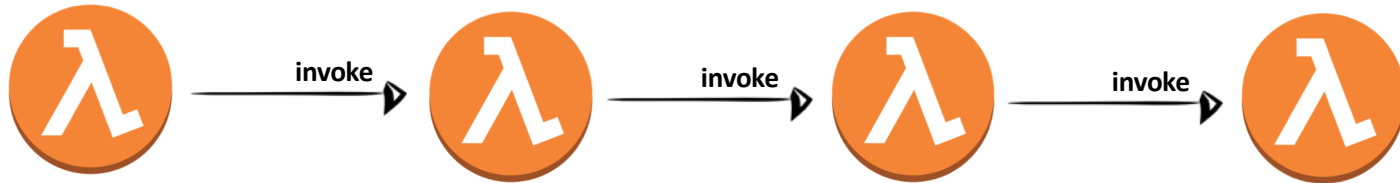


Function as a Service

Do not just lift and shift your microservices into FaaS



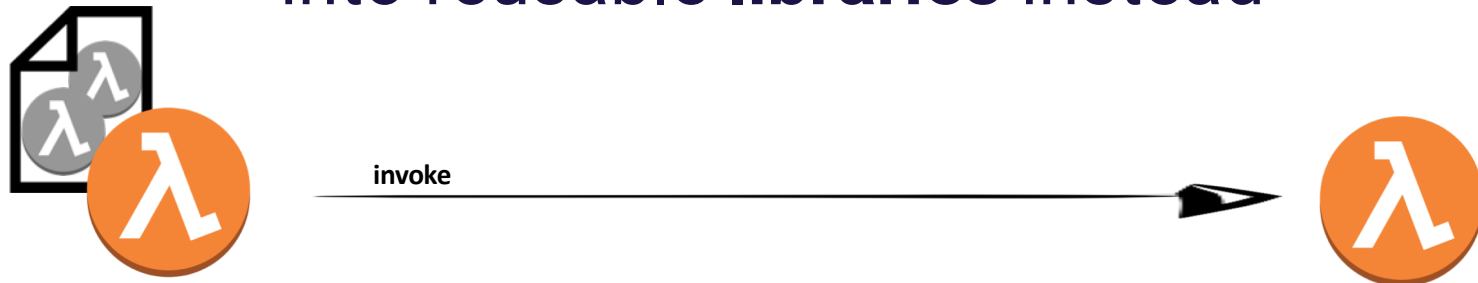
Do not end up with a collection of distributed microservices



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

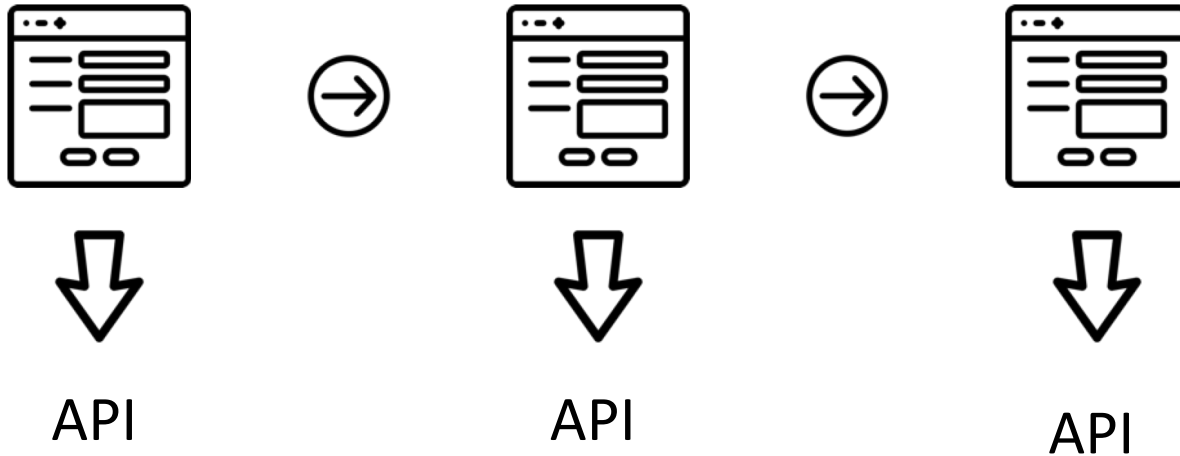


Shhhhhh.... It's ok to have more than one function in your Function as a Service...

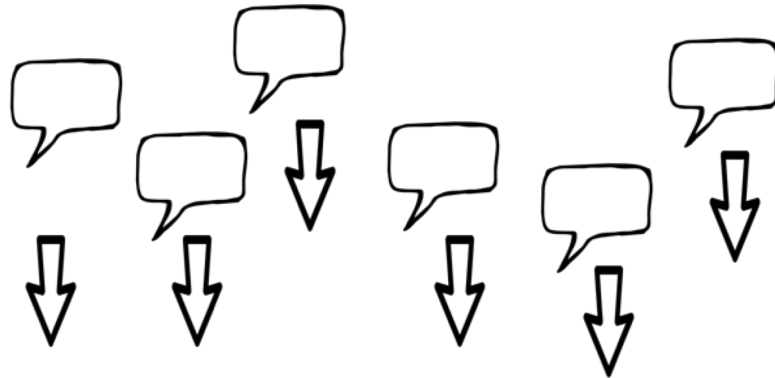
Having said that... **don't** just **lift and shift**
your microservices into FaaS



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.

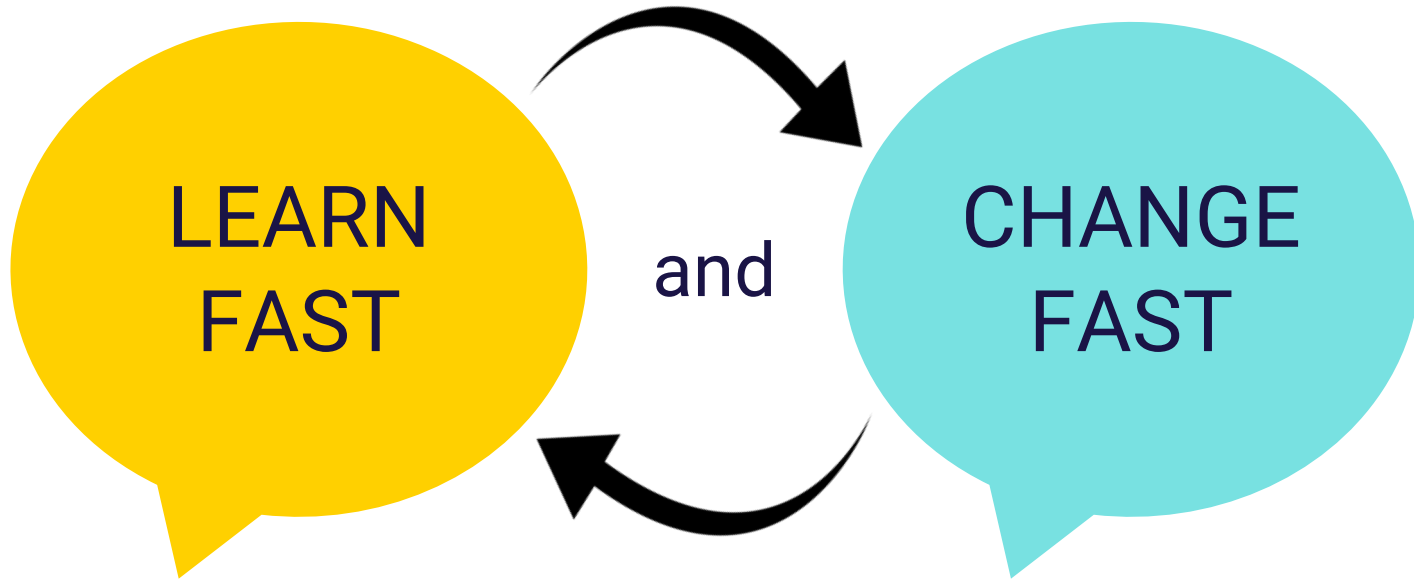


API

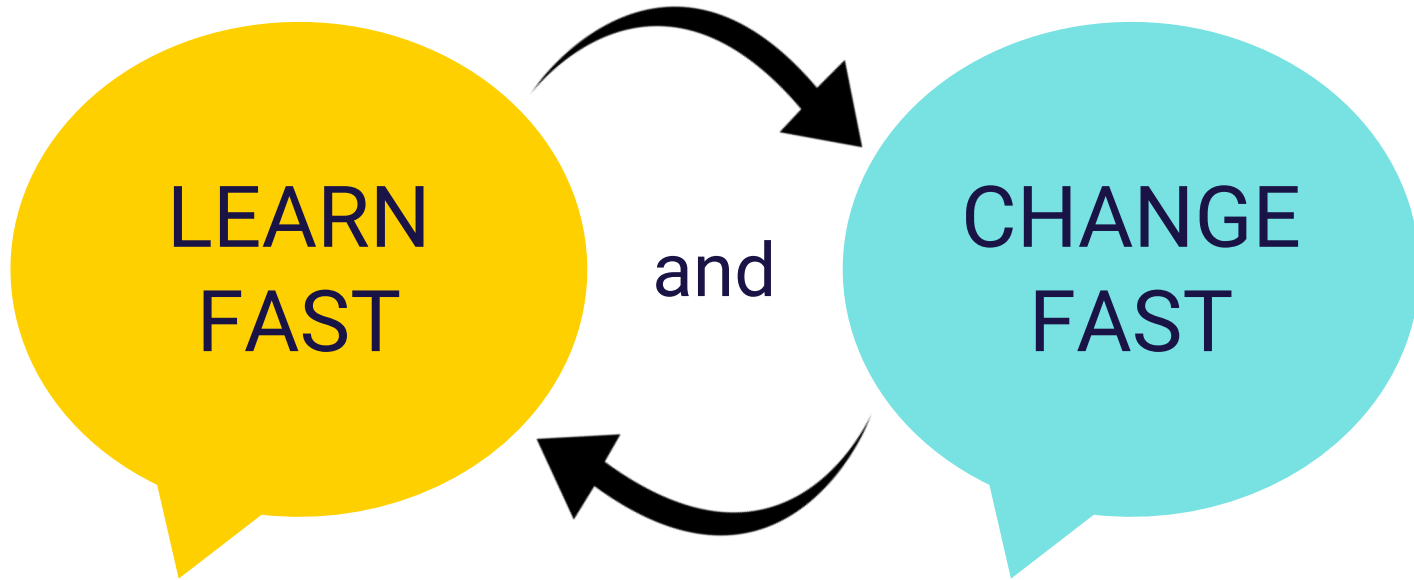
**REWRITE is not a
four-letter word**

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

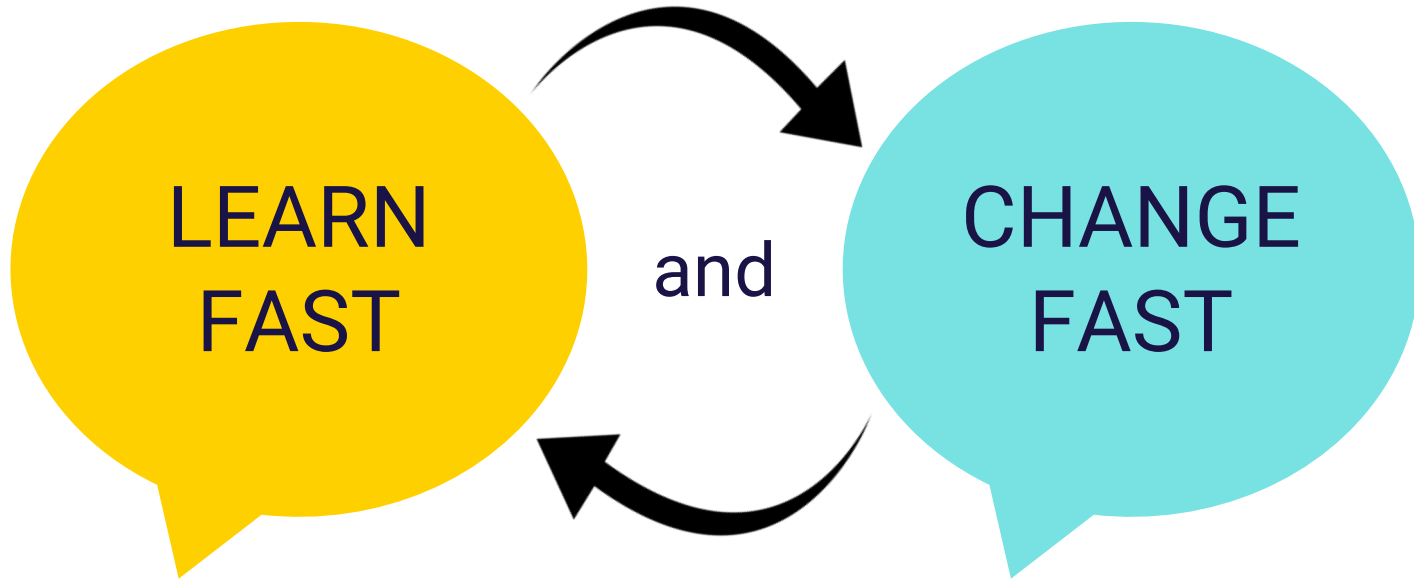
We get there **iteratively** by being able to



Both Chatbots and Serverless let us



Both Chatbots and Serverless let us



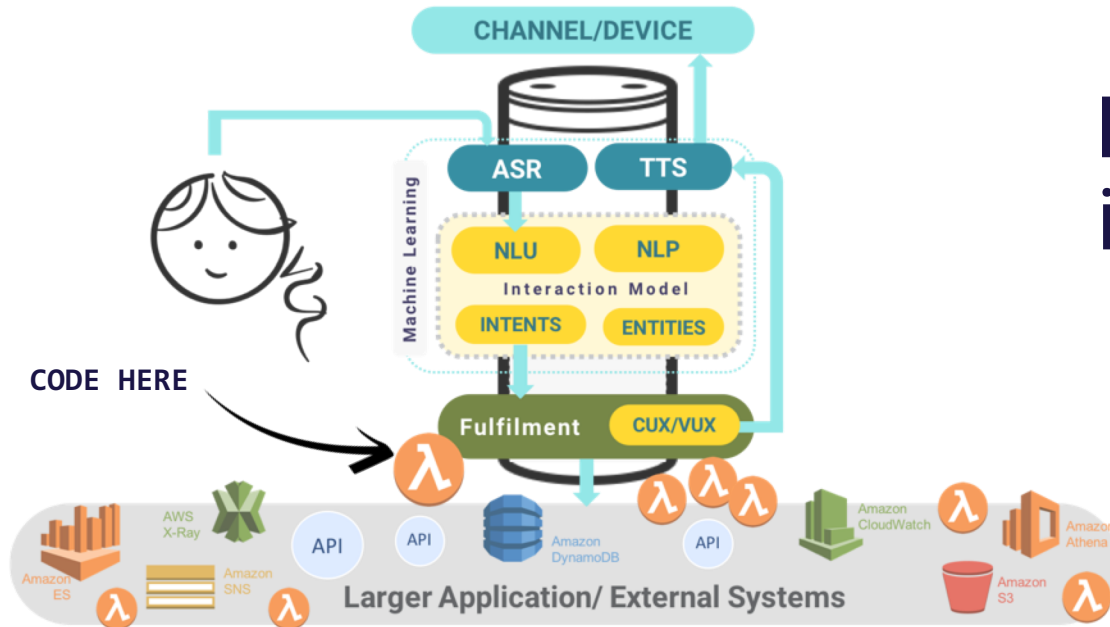
*but **Good Supporting Engineering Practices** are needed to do both **with confidence and at scale***

Amazon Lex Console

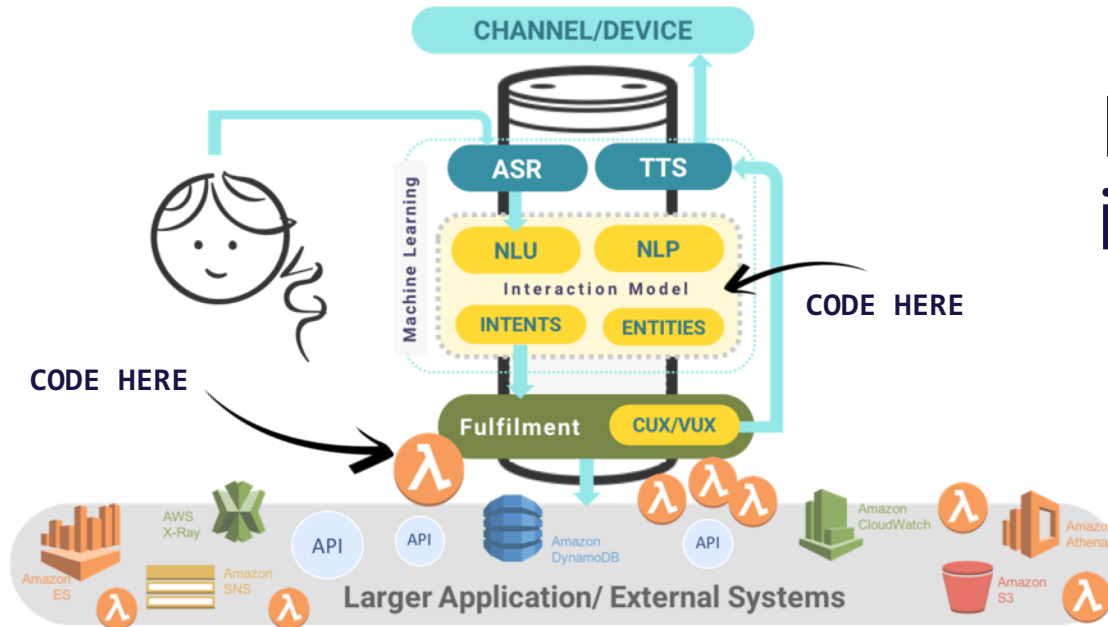
The screenshot displays the Amazon Lex console interface for a chatbot named 'DemoAssistantBot'. The 'Intents' section is active, showing the configuration for 'demo_BookMeetingIntent'. The 'Sample utterances' list includes various phrases like 'please book a meeting at (time)', 'i want to book a meeting', and 'book me a meeting (time)', with slots for 'time', 'person', and 'date' highlighted in colored boxes. The right-hand panel shows a 'Test bot' section with a 'Ready, Build complete.' status and a text input area for testing the chatbot's response.

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

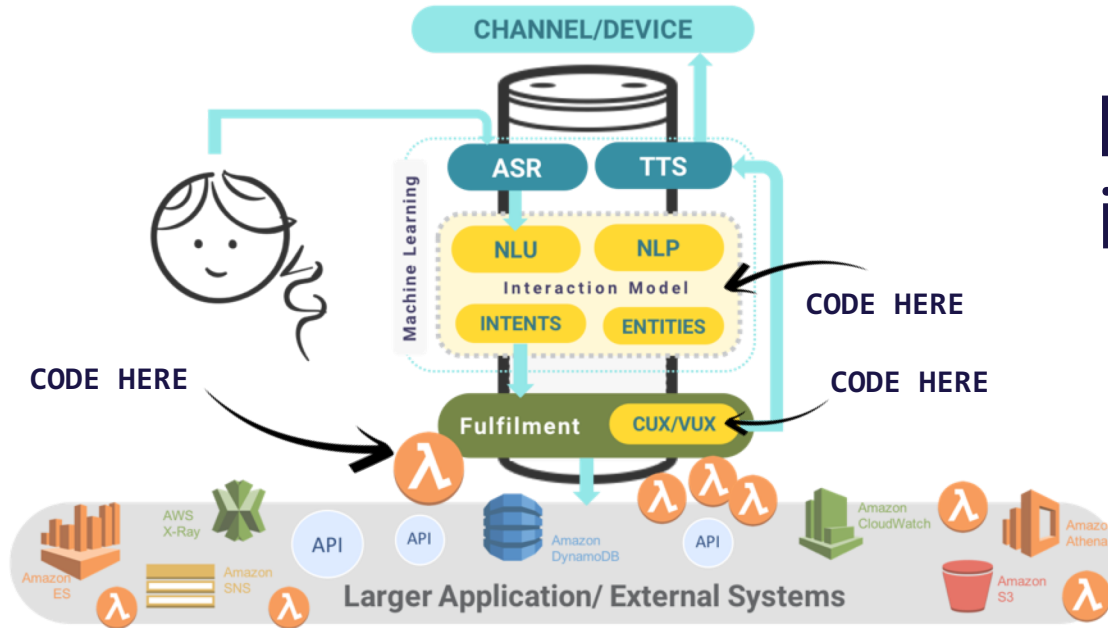
Fulfilment in Code



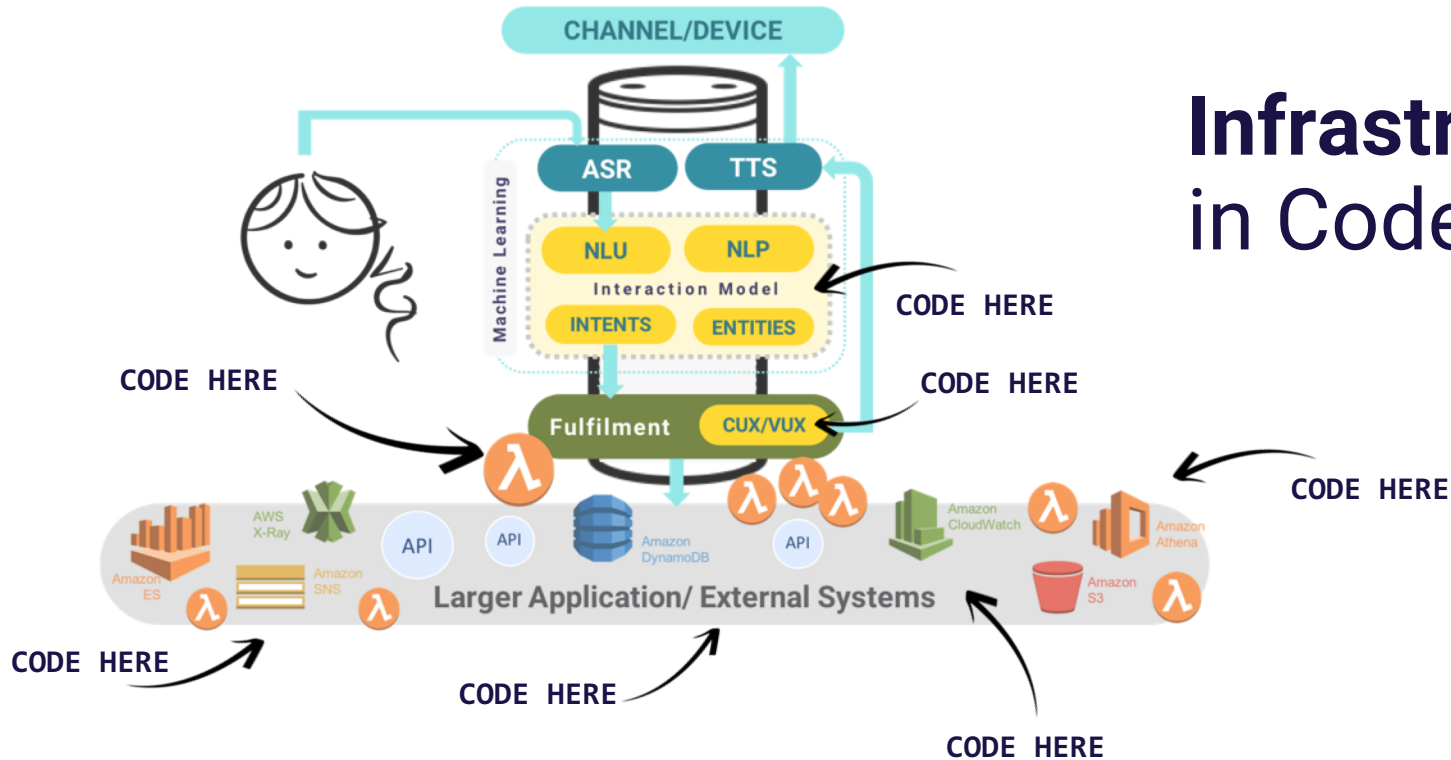
Bot Interaction Model in Code



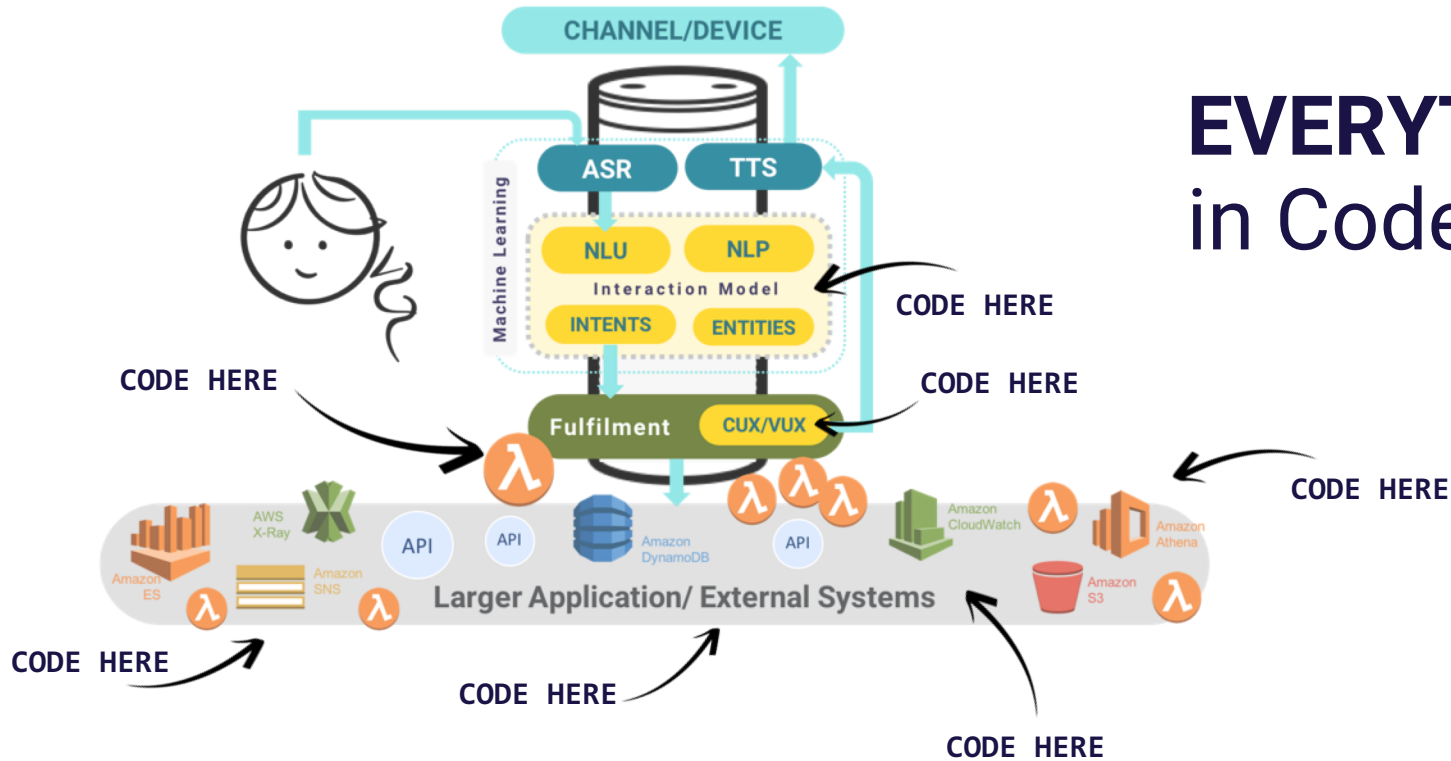
Design in Code



Infrastructure in Code



EVERYTHING in Code



- Testable
- Traceable
- Repeatable
- Observable

EVERYTHING
in Code

DemoAssistant Latest ▾

Editor Settings Channels Monitoring

Intents +

- BookMeetingIntent
- GoodbyeIntent
- GreetingIntent

Slot types +

No slots created

Error Handling

BookMeetingIntent Latest ▾

Sample utterances ⓘ

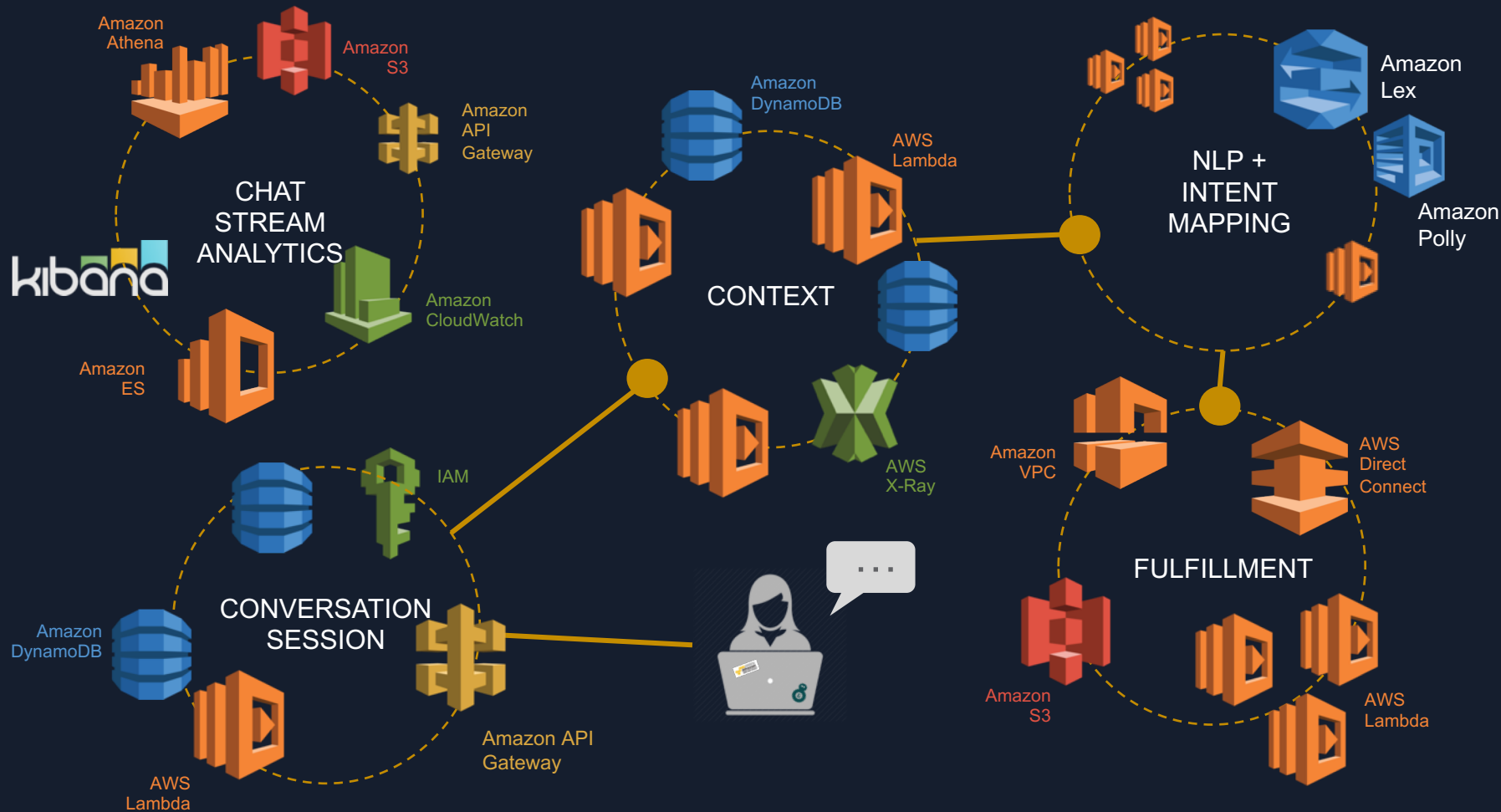
e.g. I would like to book a flight. +

- 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 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 ×

```

},
"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.TIME"
  }
]

```



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



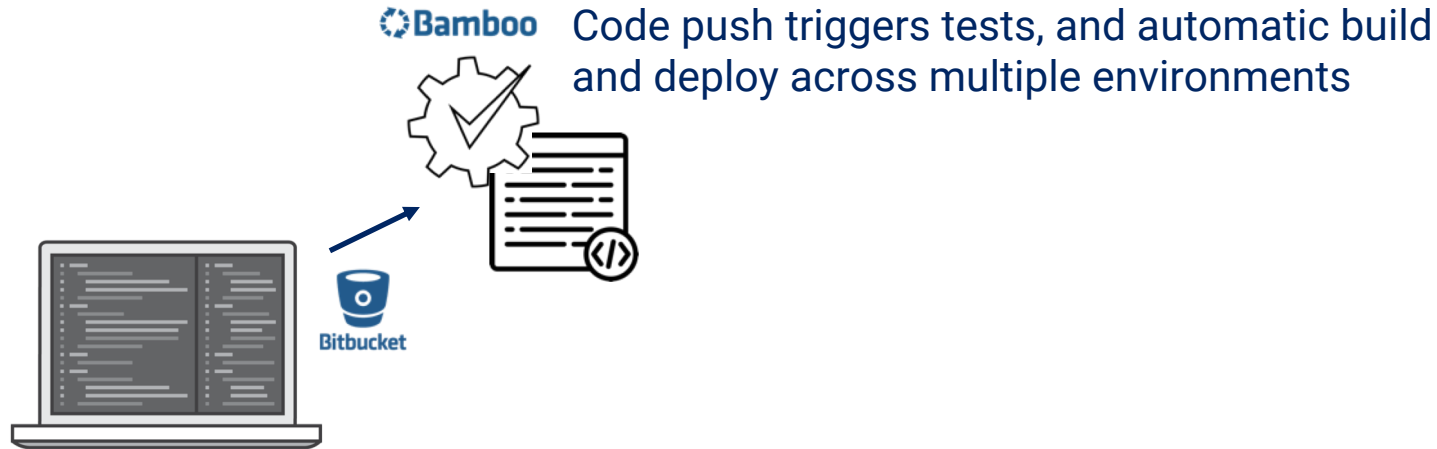
Everything written in CFN,
with JSON for API calls to
Amazon Lex Model Building
API as part of deploy

- 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

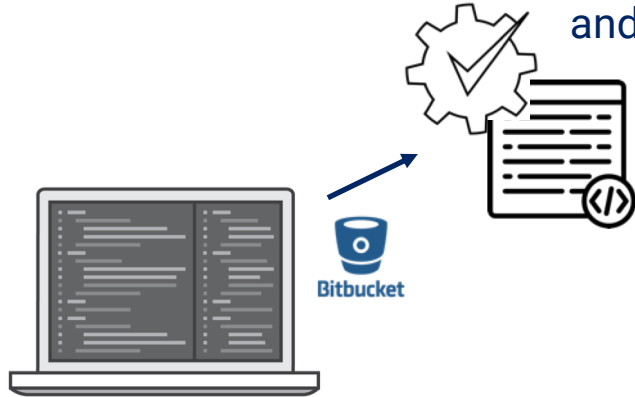


Everything written in CFN,
with JSON for API calls to
Amazon Lex Model Building
API as part of deploy

Conversation as Code - Amazon Lex

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

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




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

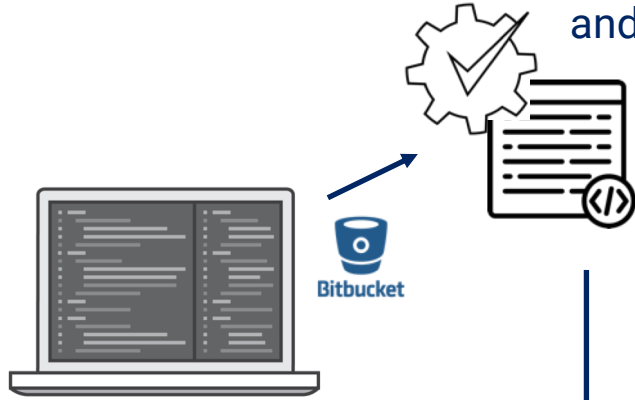
- Unit Test
- Lambdas
- Libraries
- Static analysis
- Cloud Formation
- Code
- JSON for API calls

Build Tests

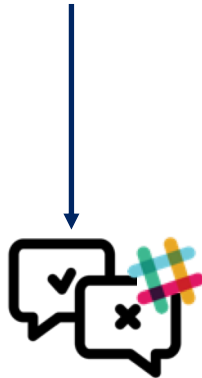
Conversation as Code - Amazon Lex

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

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



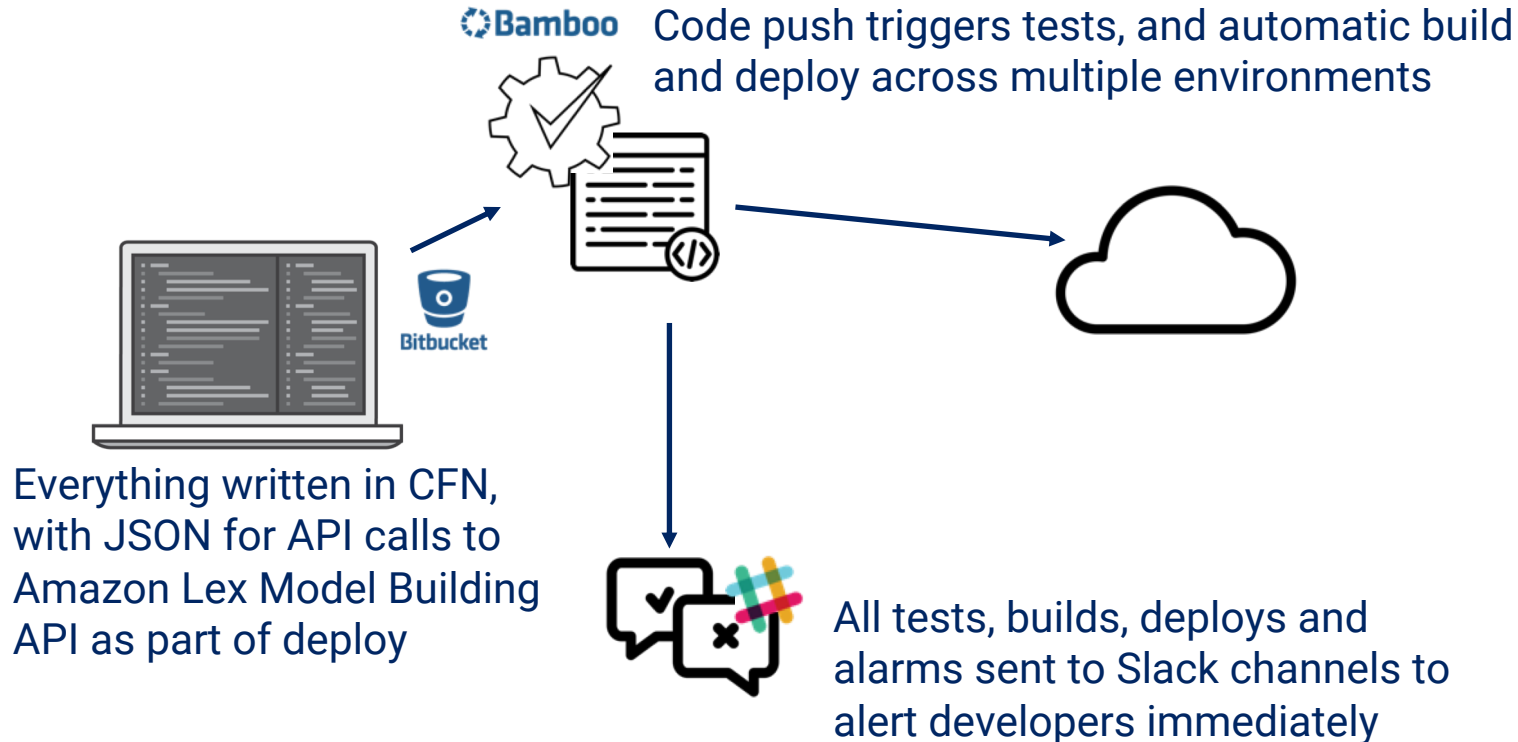
Everything written in CFN,
with JSON for API calls to
Amazon Lex Model Building
API as part of deploy



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

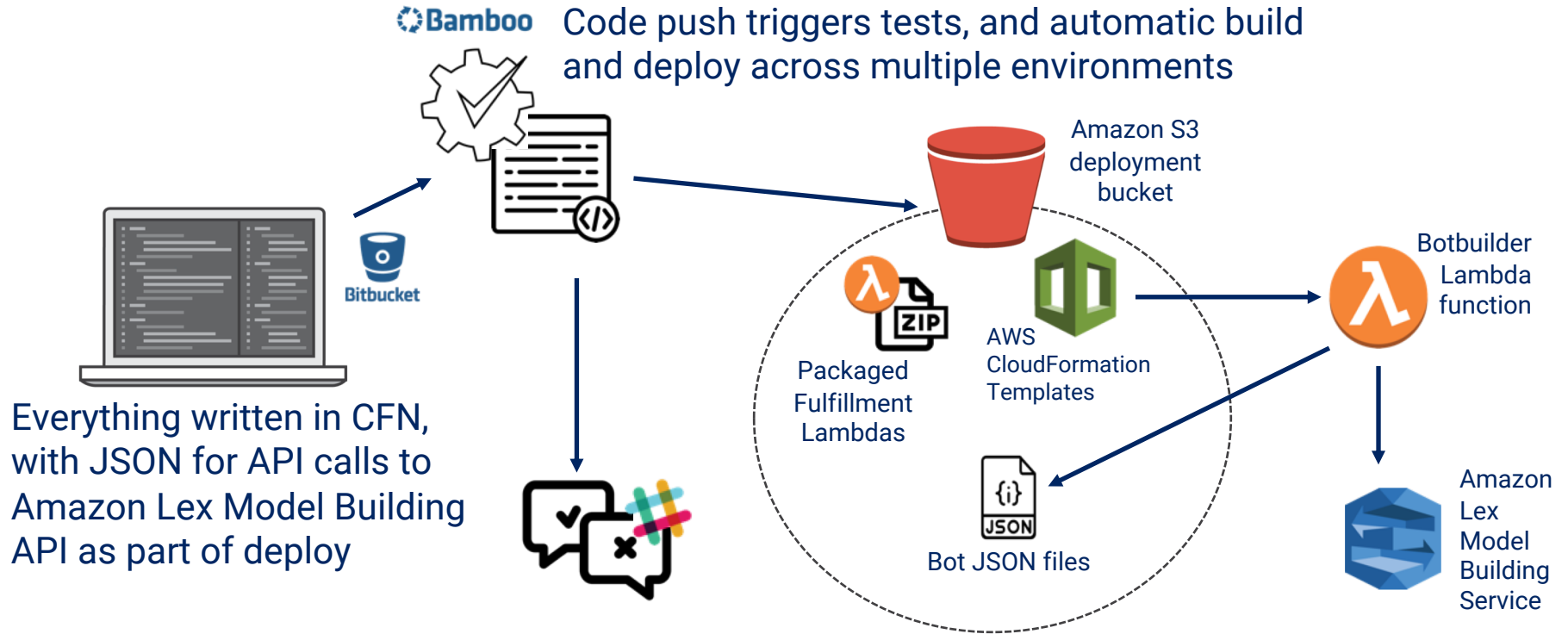
Conversation as Code - Amazon Lex

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



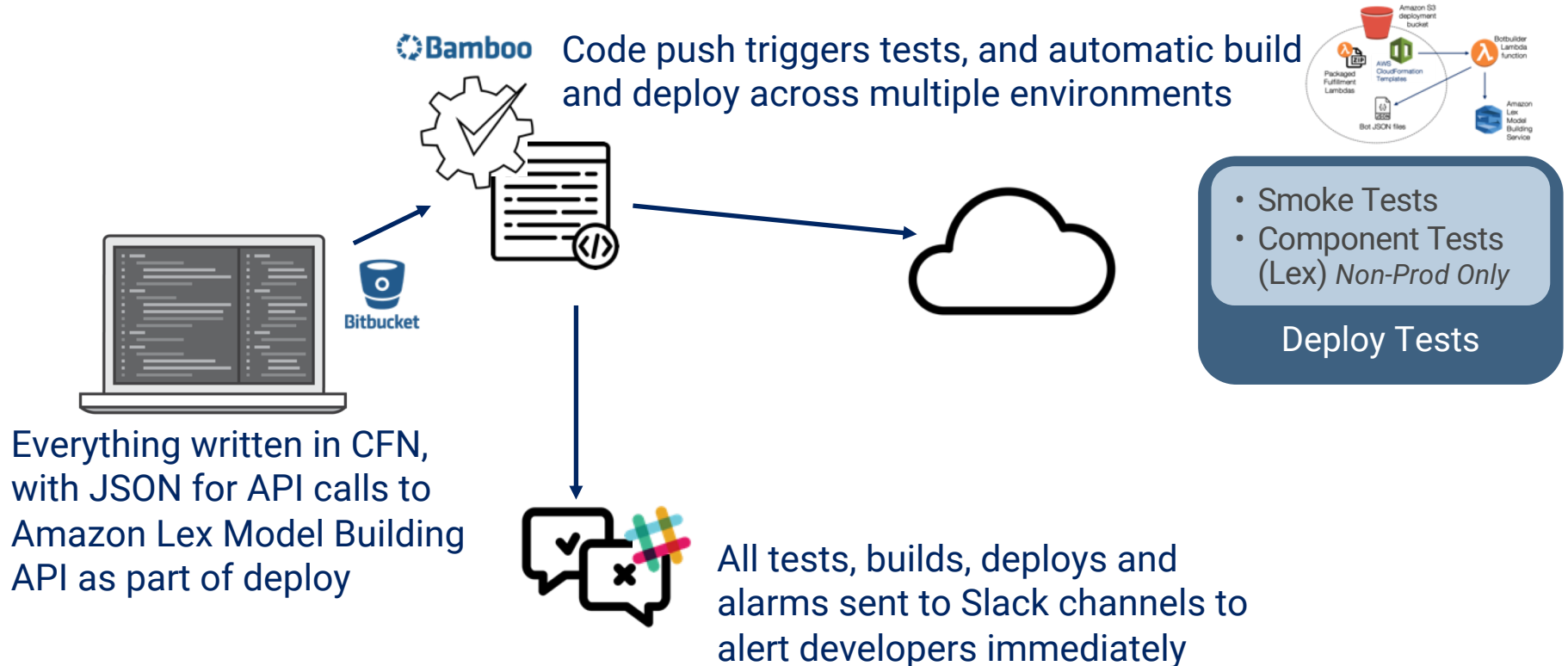
Conversation as Code - Amazon Lex

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



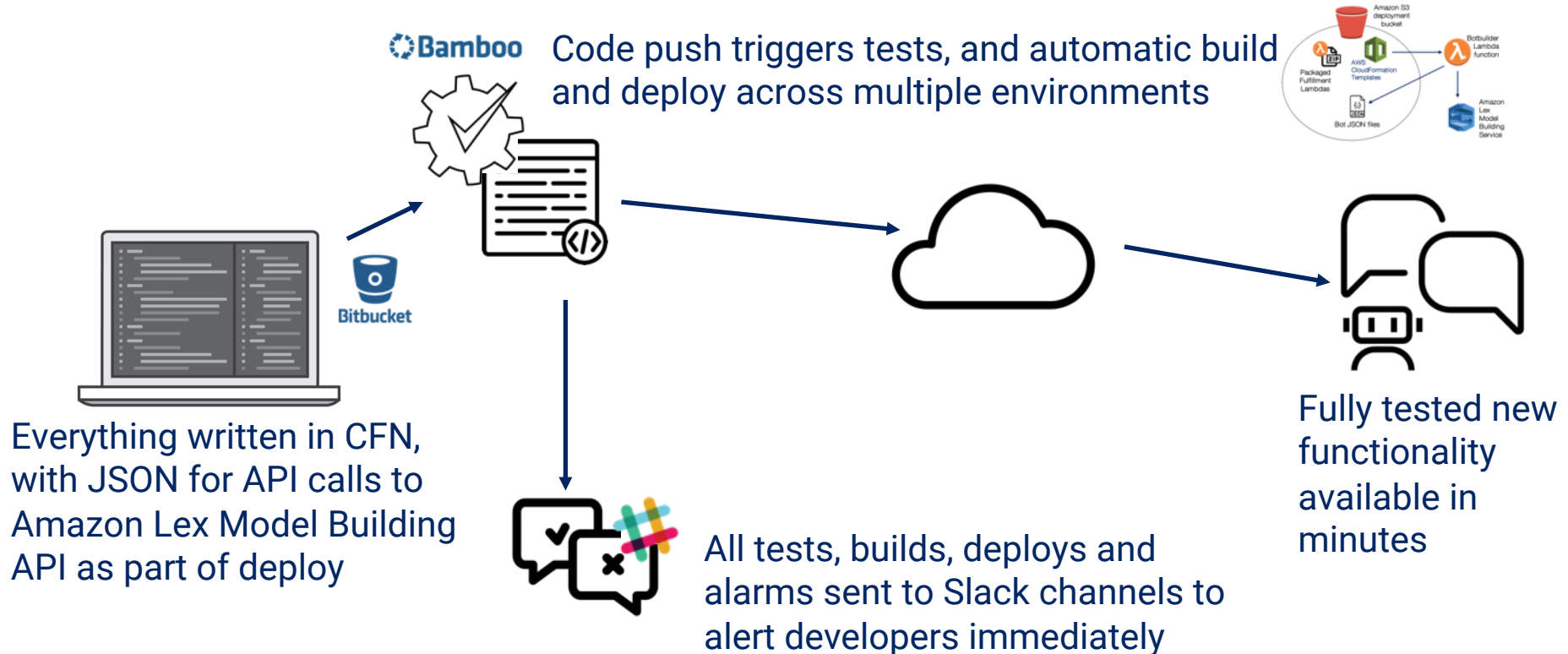
Conversation as Code - Amazon Lex

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



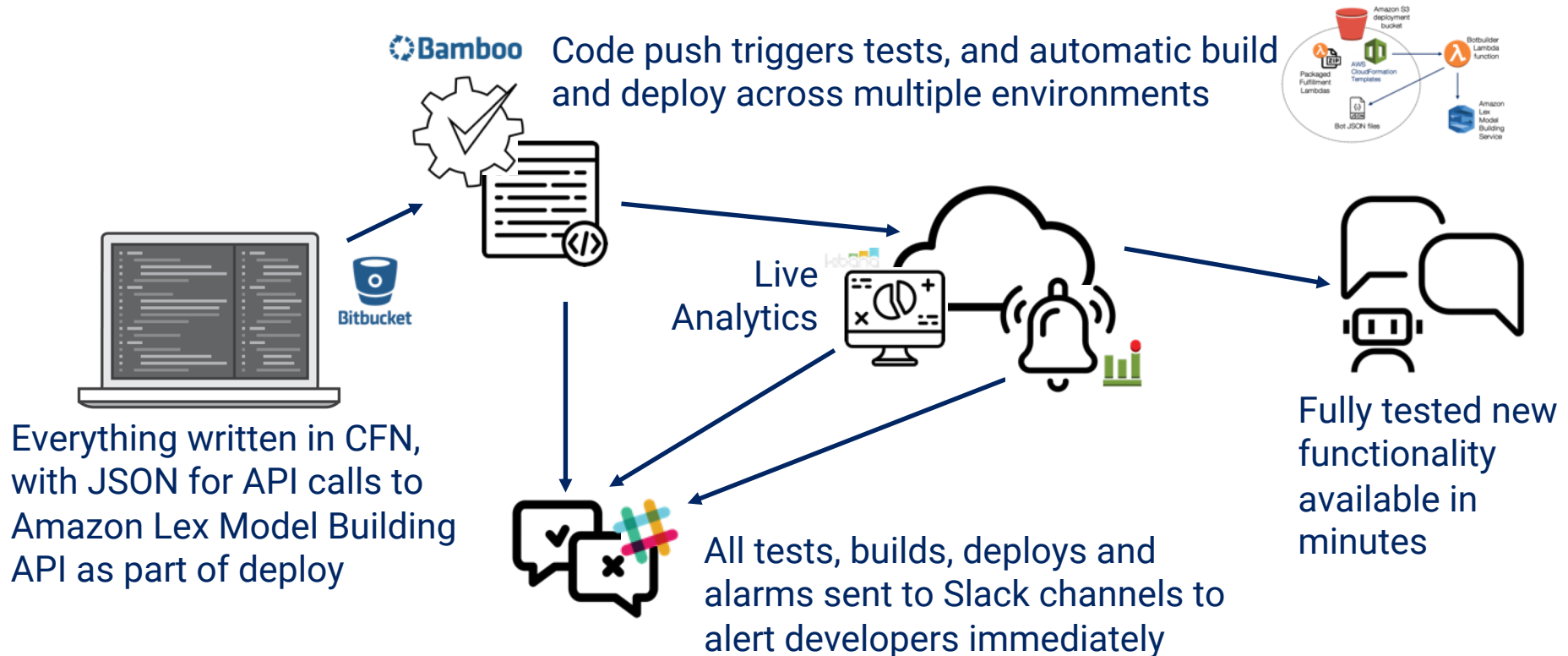
Conversation as Code - Amazon Lex

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



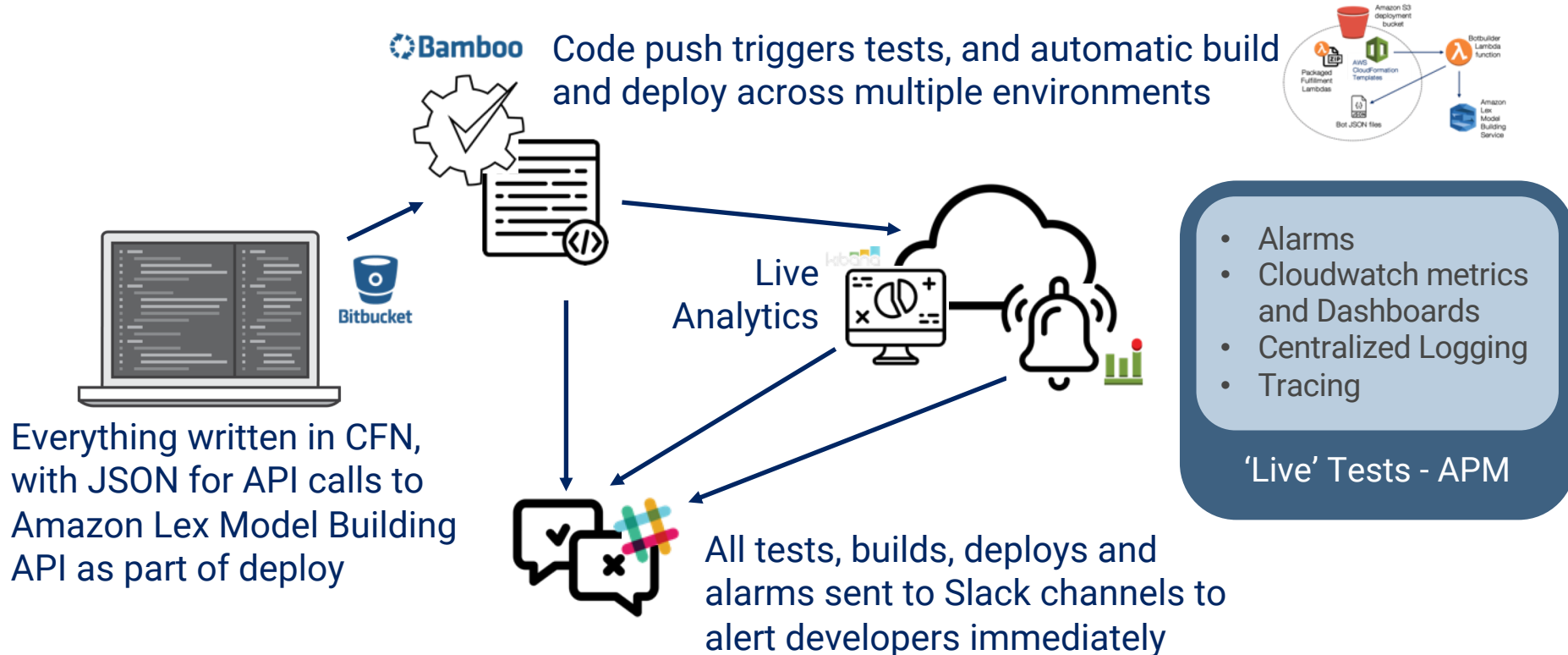
Conversation as Code - Amazon Lex

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



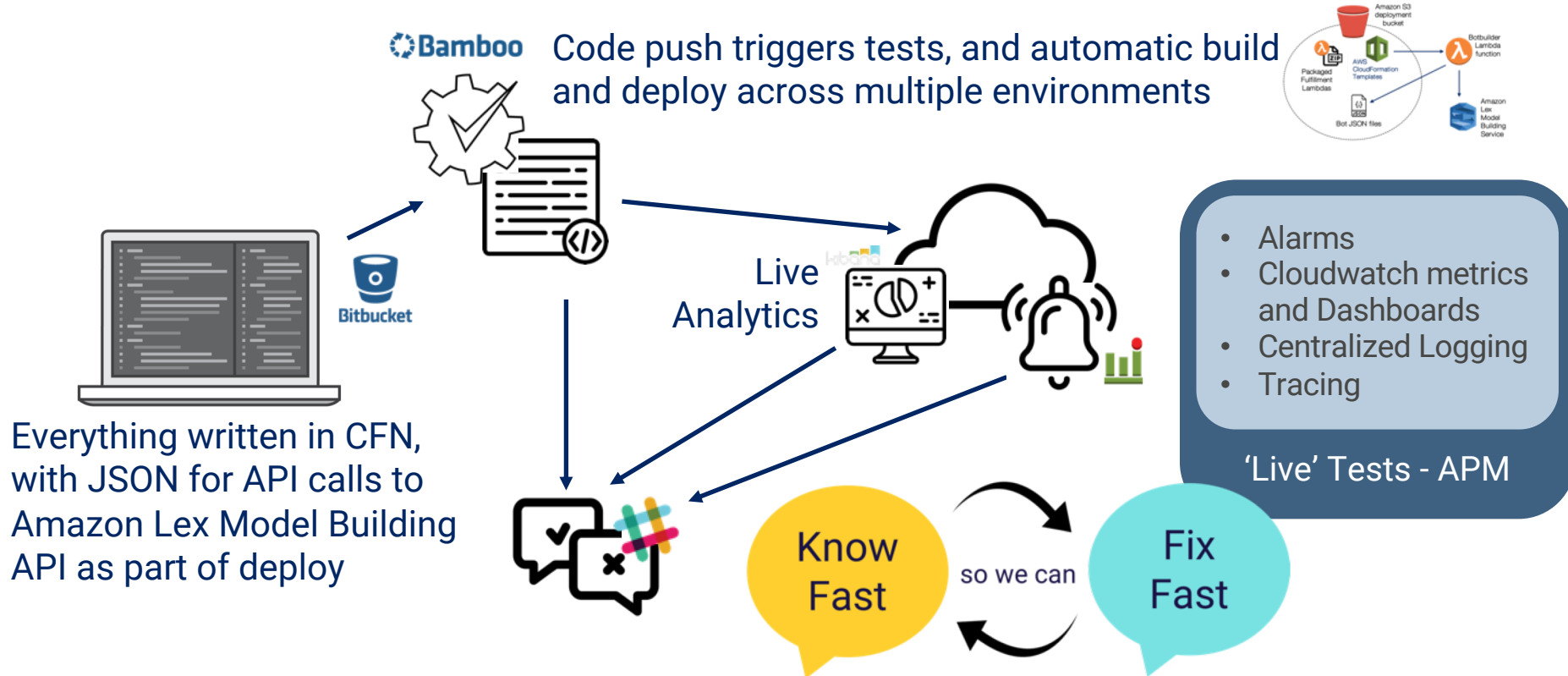
Conversation as Code - Amazon Lex

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



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.

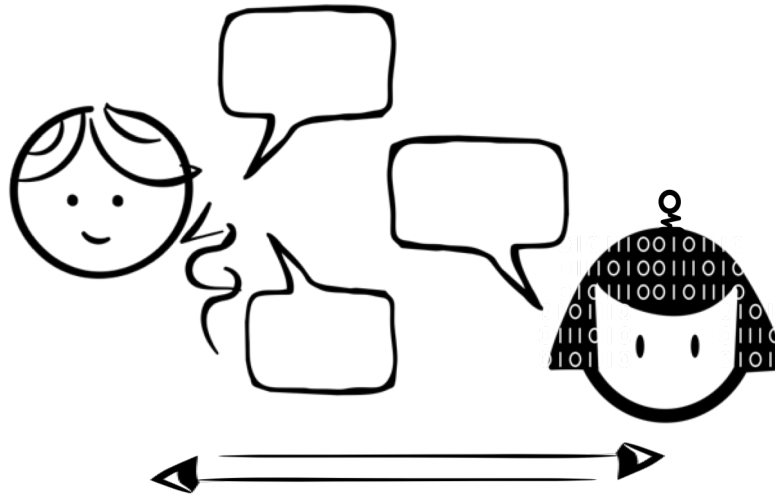
Know what isn't working and **fix** it, and keep **adding more** of what is!

Being able to

Change
Fast

lets us

Learn
Fast



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

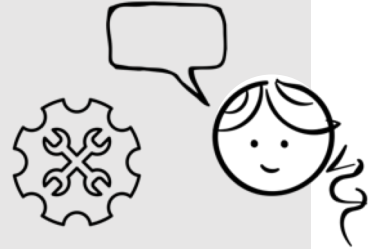
...and that's the secret to a **great chatbot!**

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>