

ChatOps的前世，今生和未来



13太保 (/u/39c54f1a1d1c) [+ 关注](#)

2017.05.09 14:27* 字数 5324 阅读 3102 评论 1 喜欢 6

(/u/39c54f1a1d1c)

在人人都在讨论 DevOps 的今天，有一个词 ChatOps 显得比较突出，什么是 ChatOps？它是怎么工作的？它能带来有什么好处？最近调研了一下ChatOps，本文将基于个人的理解，尝试回答这几个问题。

起源

ChatOps 这个词在2013年出现，最初由 GitHub 提出，他们发现在日常的工作中，需要不停的运行以下命令：

```
git checkout -b feature/xxx
# do sth.
git commit -m "Bump version"
git push origin feature/xxx
# create pull request
```

我们需要进行代码提交，确认持续集成通过，代码部署，确认各项指标正常，代码合并等一系列的繁琐工作。

于是他们开发了 Hubot 这个机器人，来帮助他们完成这一系列的工作。



创建一个Hubot

首先，我们可以通过命令行的方式与 Hubot 进行交互：

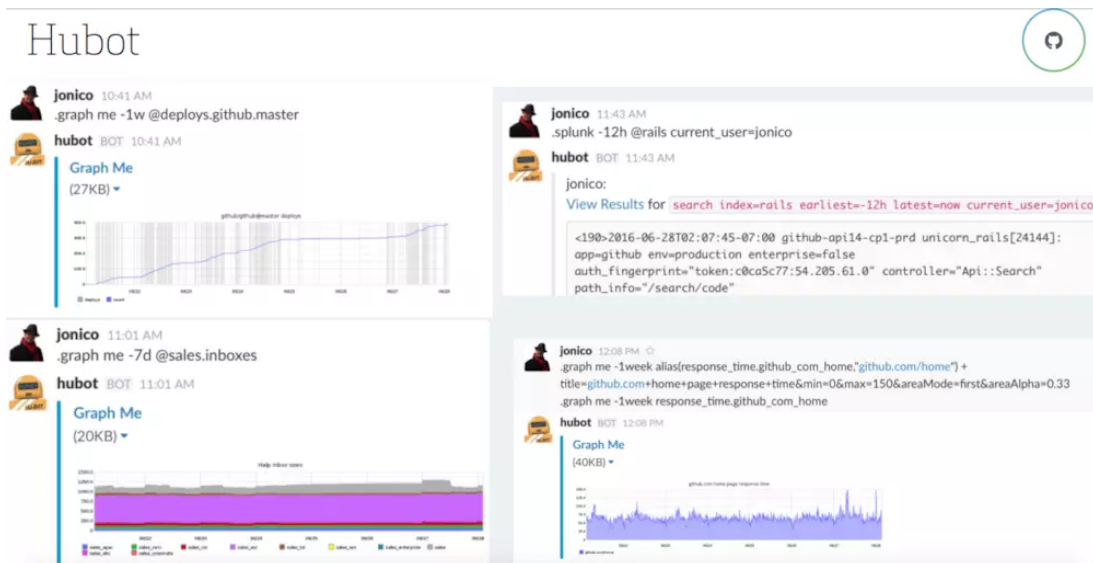
```
dakangshuji> dakangshuji ping
dakangshuji> PONG

dakangshuji> dakangshuji time
dakangshuji> Server time is: Sun May 07 2017 12:18:00 GMT+0800 (CST)

dakangshuji> dakangshuji echo "Hello world!"
dakangshuji> "Hello world!"
```

(/apps/redi
utm_sourc
banner-lic

同时，也可以通过和聊天软件集成，在聊天软件中和Hubot进行交互。它可以帮助我们执行代码部署，查看issue的状态，执行自定义的命令，甚至能直接展示监控系统的图表。



Hubot执行自定义命令

这带来了以下的好处：

- 信息透明化：一个操作能被团队的所有人看到，提高沟通效率
- 寓教于做：新的同学进入团队，能够通过观察老司机的工作方式，迅速上手
- 提升移动办公效率：出门在外，如果紧急事项需要处理，可以通过手机迅速处理

Hubot运行在node.js之上，采用 Coffee Script 编写

```
module.exports = (robot) ->
  robot.hear /badger/i, (res) ->
    res.send "Badgers? BADGERS? WE DON'T NEED NO STINKIN BADGERS"

  robot.respond /open the pod bay doors/i, (res) ->
    res.reply "I'm afraid I can't let you do that."

  robot.hear /I like pie/i, (res) ->
    res.emote "makes a freshly baked pie"
```

Hubot的理解能力比较低，采用的是 字符串匹配 的方式。例如，上面的例子中：

1. 当输入包含badger的时候，会自动发送响应 Badgers? BADGERS? WE DON'T NEED NO STINKIN BADGERS

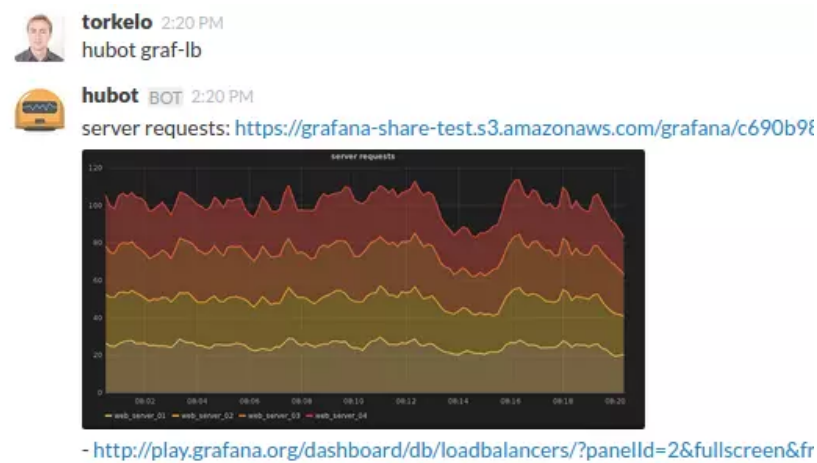


- 2. 当有人@Hubot, 严格匹配 open the pod bay doors 时, 回复发送者 "I'm afraid I can't let you do that."
- 3. 当满足条件是, 将消息群发或转发到另一个群。

Hubot支持适配到 Shell , 各种聊天工具如 iMessage , Slack , QQ , Weixin 等。

Hubot可以将自定义的逻辑部署在本地运行, 也可以部署到云端, 如 Azure , Heroku 。

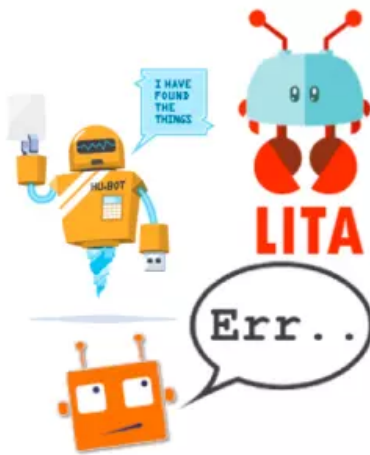
很多第三方系统, 对Hubot的支持都非常友好, 例如Graphana支持和Hubot进行交互, 给出某个指标的走势图:



Hubot+Graphana

三雄争霸

随着Hubot的兴起, 一些更多的机器人开始发展起来, 逐渐形成了 Hubot , Lita , Errbot 三雄争霸的场面:



三雄争霸

Hubot

参见上面的介绍

Lita

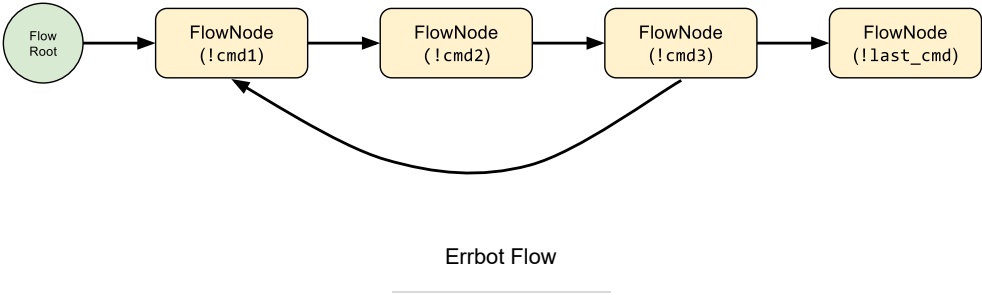


Lita沿袭了Hubot的大部分特性，但区别是Ruby编写的，底层采用了Redis做存储。

Errbot

Python编写，支持和主流的聊天工具如Slack, Telegram等进行集成

Errbot引入了一个 workflow (Flow)的概念，使得在聊天软件中进行流程化操作成为可能，例如权限审批等操作。



百花齐放

随着各种 ChatBot 的兴起，各大聊天工具坐不住了，两大主流的聊天软件， Slack 推出了自己 Slackbot 。

Slackbot

在SlackBot里面，提供了内置的一些机器人，同时开放了接口，允许第三方自由定制自己的机器人。例如，可以设置提醒，看天气：



Slack Bot

查看会议室空闲情况：

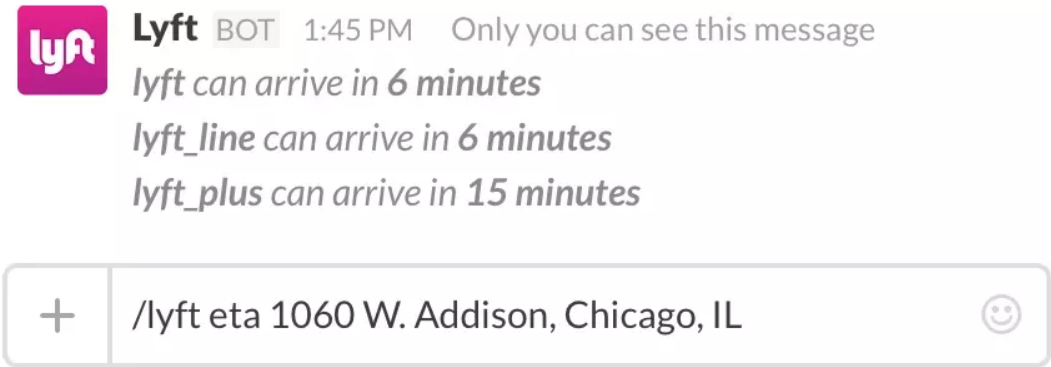


 Zach Dunn 12:22 PM

查看会议室

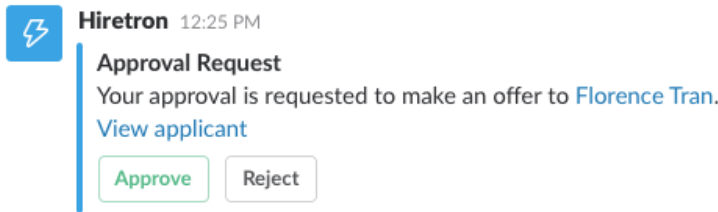
甚至集成了叫车服务，可以直接在聊天窗口里面叫车！

(/apps/redi
utm_sourc
banner-clic



SlackBot叫车

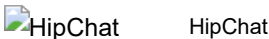
进行权限审批，在APP里可以通过按钮进行交互：



在聊天窗口内进行权限审批

HipChat

HipChat 也推出了自己的ChatBot平台，集成了众多的ChatBot。



Facebook Messenger

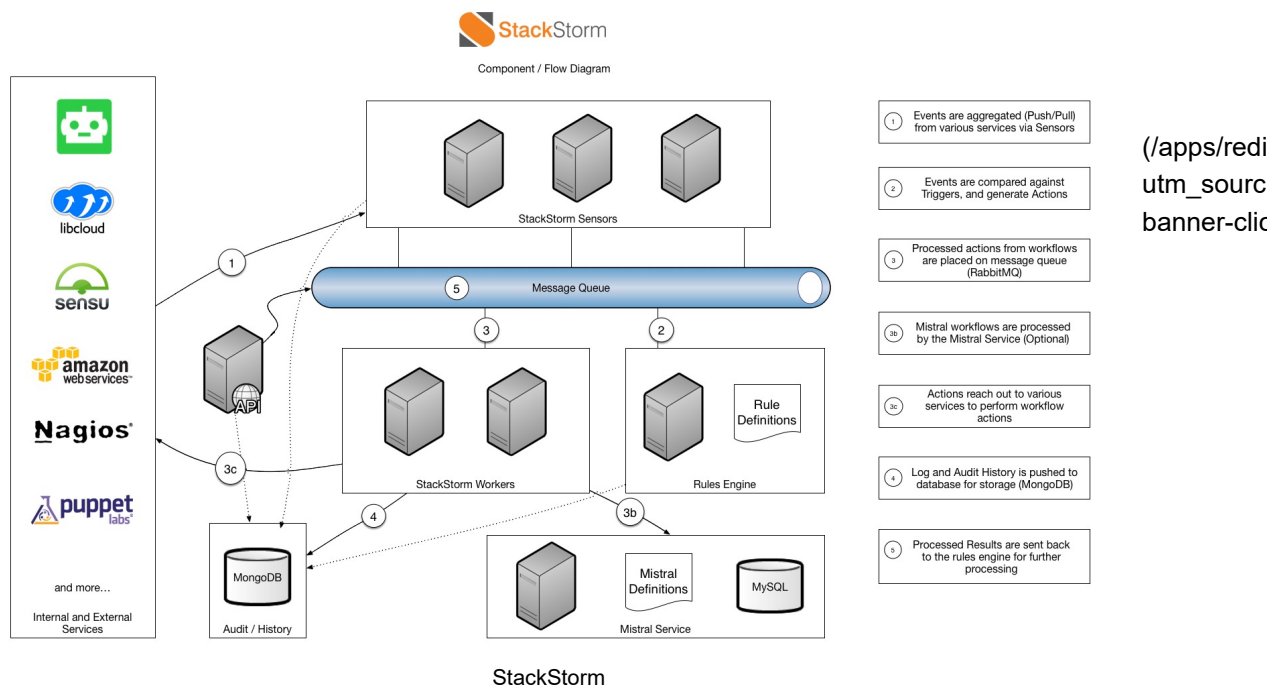
而Facebook也宣布旗下的Messenger应用开始接受用户开发的聊天机器人。

第三方生态

StackStorm (<https://link.jianshu.com?t=https://stackstorm.com/>)

StackStorm是一个基于事件驱动的自动化平台，通过一些sensor感知到事件，经过规则匹配，触发action的执行，并返回结果。而chatops只是它的一个应用场景，通过和聊天软件的集成，将ChatOps中的逻辑封装在规则校验和执行动作中，它目前只能执行一些python或者Shell脚本。





Deploybot (<https://link.jianshu.com?>

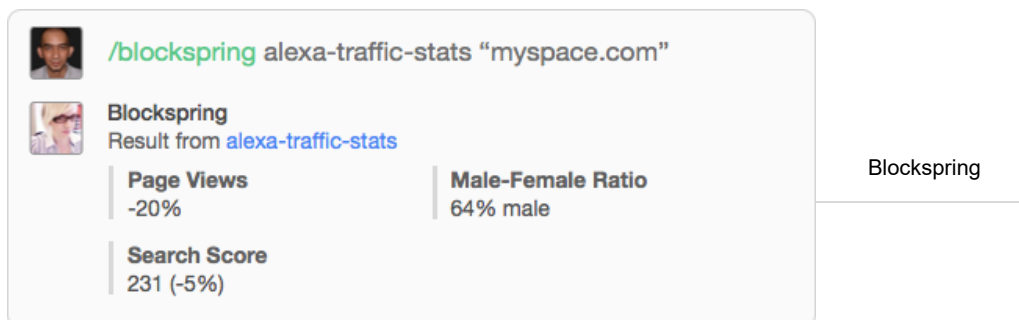
t=<http://deploybot.com/guides/chatops-deploybot-slack>)

Deploybot 专注于代码的部署上，可以通过聊天工具在命令中将进行代码的部署并查看部署的结果

Blockspring (<https://link.jianshu.com?>

t=<https://www.blockspring.com/slack>)

blockspring是基于Slack发展出的生态，能够通过聊天工具进行员工信息查询，和社交网站进行交互等操作。

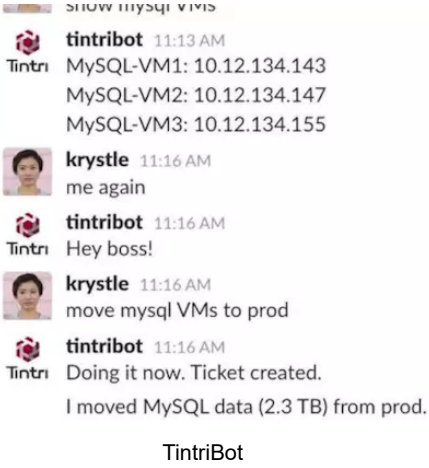
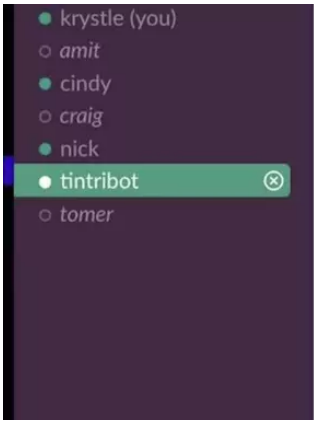


从官网上给的例子来看，都比较基础，个人感觉使用实用不大。

Tintri

这家公司很有意思，他们为Slack的Tintri能够通过聊天机器人中进行硬件资源的管理，例如弹性扩容，数据拷贝等工作。





(/apps/redi
utm_sourc
banner-clic

WebTask

WebTask是由 Auth0 这家公司推出的一项方案，当初的目的旨在结局多租户环境下不同用户的代码如何进行隔离，因此提出了Code Sandbox这一个概念。

但其实它却由此延伸出了另一个重要场景 WebHook ，运行快速的发布和运行一段业务逻辑，把代码从服务器中抽取出来，代码其实成为了这个场景下的一个请求参数。

WebTask 支持Node.js和C#，通过 EDGE.JS 实现了异步调用C#的lambda表达式。

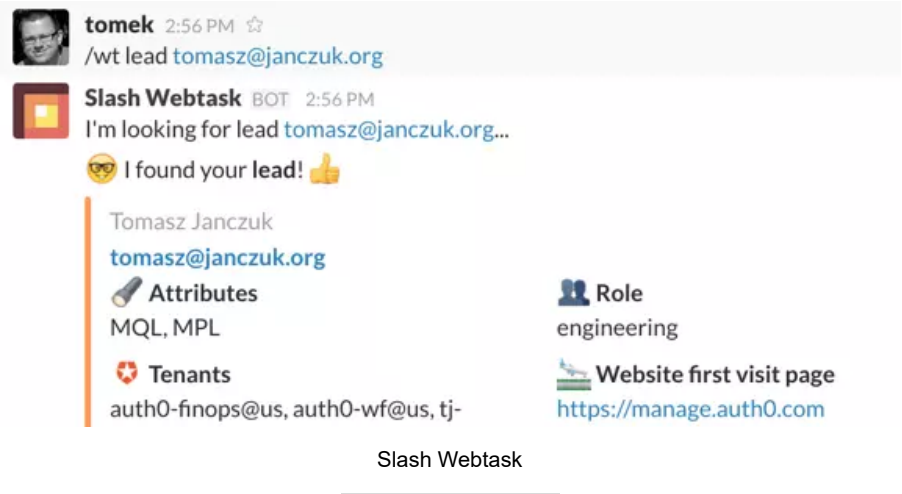
核心概念：

- web task token ：标示一段后端执行逻辑(代码)+加一段秘钥。这段逻辑可以存储在一个第三方系统。这时候后端系统已经变成了一个业务无关的单纯的执行引擎，然后客户端只需要发送这个token给后端系统，后端系统会获取这块代码，执行，并返回结果。
- web task request ：表示包含了web task token的一个请求
- web task runtime ：通用的能执行webtask的运行时。



下面这个例子中，Webtask通过和Slack集成，实现了查看leader信息的功能





(/apps/redi
utm_sourc
banner-clic

WebTask和JStorm之类的实时处理引擎有什么区别？个人认为主要还是 快速 和 轻量，JStorm是 7*24持续运行 的，WebTask基于 事件触发，用完即停止。

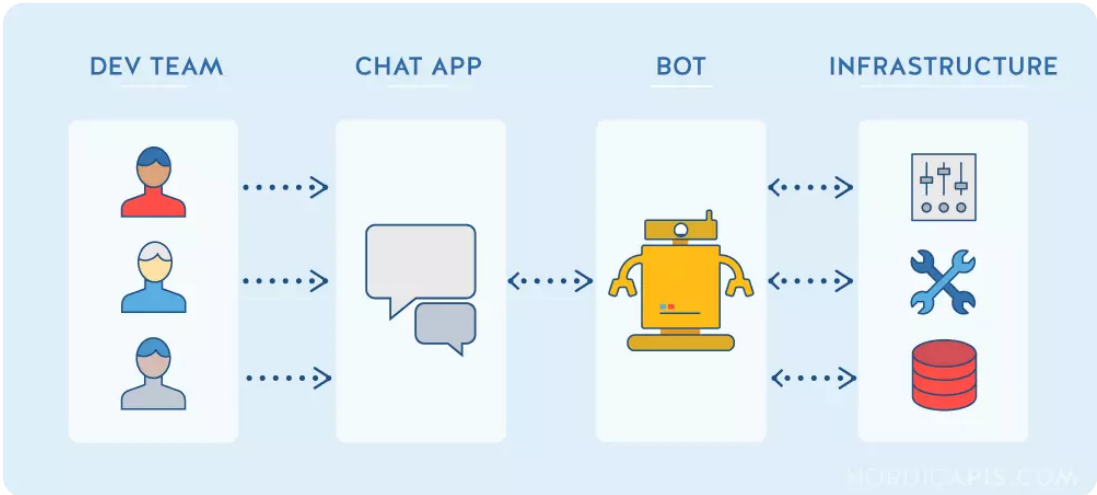
小结

可以看到，从 HuBot，Lita 到 SlackBot，基本上用户在和聊天机器人的交互过程中，还停留在命令式的交互阶段，即使有自然语言，也是非常有限几个的单词，聊天机器人也只能通过 字符串匹配 这样比较朴素的方式来试图了解用户的意图。因此，我把这个阶段定义为ChatOps的初级阶段，可以称为 CommandOps。

在这种模式下，聊天软件相当于传统命令行下的 Terminal，通过输入命令，返回结果。但它和Terminal的显著区别在于两点：

- 1. 共享化。这个 Terminal 可以是群里面的成员都能看到的，接触群聊功能，它极大提高了信息沟通的效率。尤其是需要团队协作完成一件事情的时候。试想一下，我们在沟通的过程中，当需要在群里向别人表达一个意图的时候，有没有打开一个网站，或者是在在某个页面上进行一番操作，然后截图贴给对方？
- 2. 移动化。随时随地能够通过移动APP和机器人沟通，让机器人完成本来在命令行，或者是web端才能完成的任务。目前公司内的各种ops，几乎没有多少是针对手机优化过的。因此很多同学都有过这样的经历，出门在外还必须带个电脑，因为很多操作手机上根本没法完成。这里并不是说100%完全能解决问题。

技术架构



ChatOps的核心在于，把之前web端，命令行下通过人工进行的操作，通过聊天工具用机器人来代替。

ChatOps的前提是，所有的后端服务，包括基础设施，都必须通过API进行暴露，在微服务日益流行的今天，几乎所有的应用都会暴露自己的API，不管是HTTP，还是远程方法调用。如果你的应用只能和web页面强绑定，那么个人认为你的架构是落后的。

同样，我们也可以看到，在这个阶段，机器人还无法完整理解用户的语义，而随着人工智能，特别是自然语言处理技术的成熟，ChatOps也开始演进到下一个阶段，我把它称为 NLUops (Natural Language Understanding)。

Natural Language Understanding

Amazon Lex

最近的一条新闻，Amazon Lex正式发布 (<https://link.jianshu.com?t=http://www.infoq.com/cn/news/2017/04/amazon-lex-generally-available>)，吸引了大家的注意：

Amazon Lex 是一种使用语音和文本在任何应用程序中构建对话界面的服务。Amazon Lex 提供高级的自动语音识别 (ASR) 深度学习功能，可以将语音转换为文本，还提供自然语言理解 (NLU) 功能，可以识别文本的意图，让您能够构建用户体验极具吸引力且会话交互逼真的应用程序。借助 Amazon Lex，支持 Amazon Alexa 的深度学习技术现在可供所有开发人员使用，从而使您能够轻松快速地构建出掌握自然语言的精密对话机器人 (“聊天机器人”)。

其核心概念

- **意图 (Intent)** 。意图表示用户想达到的目标，可以是获得一个问题的解答，或是在远程服务上执行一个动作。
- **表述 (Utterance)** 。表述是关联到各种意图的短语。表述可以看成一种语句模板，其中可包括称为“数据槽位” (Slot) 的占位符。在一个具体表述中，数据槽位值是由用户提供的。
- **数据槽位 (Slot)** 。正如上面所提及的，数据槽位提供了一种对表述输入的表示机制。每个数据槽位代表了一种特定类型的信息，例如数量、年份、国家、城市等。也可以根据对输入的处理需求而自定义用户数据槽位类型，例如动作列表、颜色列表等。
- **提示 (Prompt)** 。提示是Lex向用户提出的问题，让用户进一步提供一些在最初的表述中并未给出的信息片段。要实现用户与使用Lex的机器人间进行多轮口头交谈，提示是一个基础部分。
- **实现 (Fulfillment)** ， 对于负责实现用户意图的AWS Lambda服务，Amazon选用了“实现”一词表示。在业务逻辑上，“实现”依赖于Lex给出的意图，而意图是根据用户表述以及表述中数据槽位的实际值识别的。



如图所示，预定鲜花是用户的意图。

I'd like to order flowers 是一种表述 Utterances , Lex发现这个表述里面缺少一些 Slot, 比如什么类型的花? 送到哪里? 这时, 它就会通过Prompt来询问用户, 当信息补充完整之后, 会调用Fulfillment。这里的只是简单的本文响应。

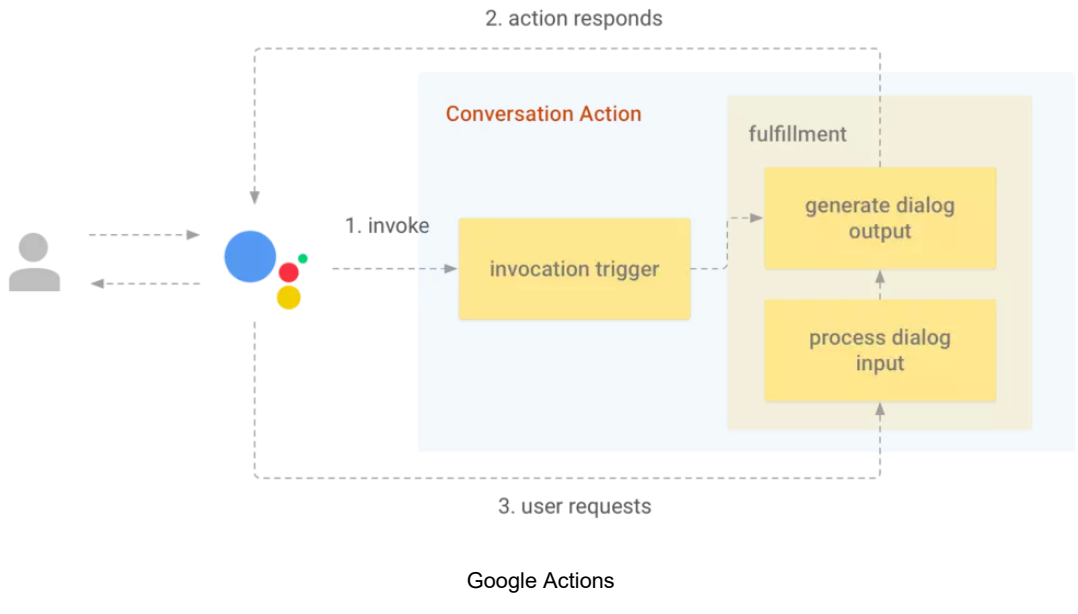
实际场景种的Fulfillment肯定不会如此简单, 通常需要一些业务逻辑的处理, 而Lex强大的地方在于, 它有一个强大的后台AWS Lambda, 能够直接托管这些业务逻辑, 而用户无需关系代码逻辑在哪里运行。简直就是天生一对嘛!

Google Actions & API.AI

随着Amazon在去年的re:Invent大会上宣布Lex发布, Google紧接着也公布了自家的 Google Actions (<https://link.jianshu.com?t=http://www.infoq.com/cn/news/2017/01/Google-AI-API>), 允许开发者能够构建基于 Google Assistant的语音交互应用, 并与Google Home设备进行集成。

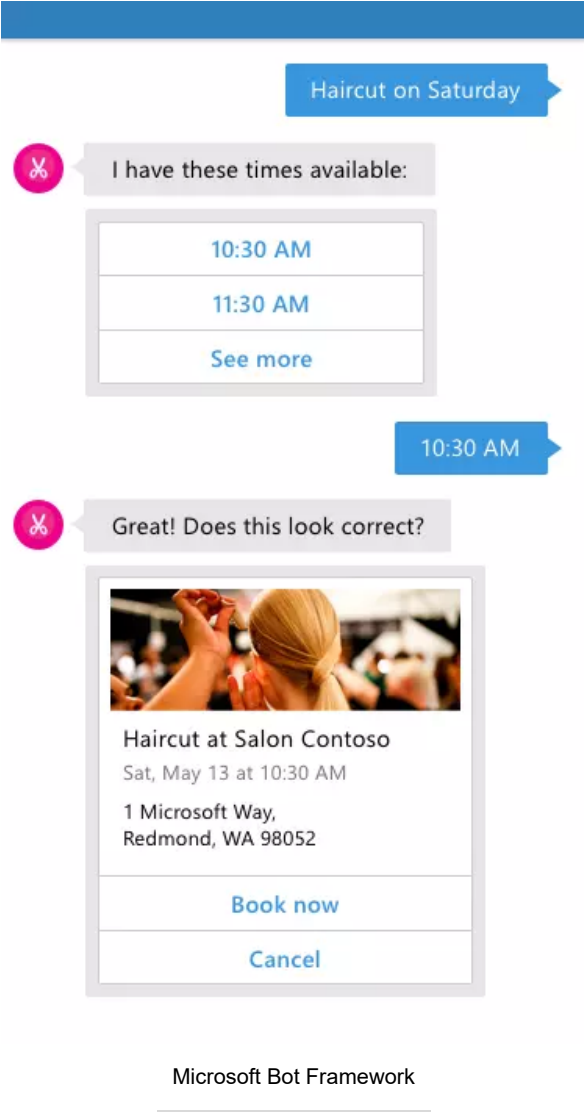
核心概念:

- 调用触发器 (Invocation Triggers) 定义了用户如何调用和发现Action。一旦被触发, Action就会展开一个由对话框定义的会话与用户交互。
- 对话框 (Dialogs) 定义了用户如何与Action进行交谈, 同时它也是Action的用户界面。它们依靠实现代码推动会话的前行。
- 实现 (Fulfillment) 是处理用户输入并返回响应的代码, 它是一个REST服务。实现通常还包含一些执行具体行为的逻辑。



调用会话行为的方式有三种:

- Conversation APIs, 它提供了与Google Assistant通信所必须使用的请求和响应的格式。



(/apps/redi
utm_sourc
banner-clic

微软这套框架提供了C#和NodeJs的SDK，但是没有API.AI那样的图形界面帮助用户快速开发。

后端服务可以托管在微软自家的Servless框架`Azure Bot Service` (<https://link.jianshu.com?t=https://azure.microsoft.com/en-us/services/bot-service/>), 提供pay as you go的体验。

小结

在这一阶段，我们可以看到，各大公司都已经看到通过机器人来达到更好的交互体验，这一块具有极好的发展前景。因此巨头们纷纷开始入场，争先恐后的开始抢夺市场。

但是，巨头们提供的平台也好，框架也好，绝大部分都是通用的框架，解决的还是日常生活中的普通场景，如订花，做菜等等。对于Ops这个特殊场景来说，其实个人认为目前的理解能力是不够的，这块还完全处于 新兴阶段 。例如，对于Ops来说，机器人如何理解机器，应用，服务，这些实体概念，还需要框架层面的深度学习才能正确理解。

通过对这一阶段的调研，我对ChatOps的理解有发生了一些变化。之前我认为，ChatOps的Chat，对应的就是聊天软件，因此ChatOps就是依托在聊天软件之上的。但现在我认识到，ChatOps的Chat其实就是一个交互式的聊天体验，只要你能友好的交互



方式，在任何的场景下都能和机器人进行沟通，让机器人理解并响应你的Operation，这其实就是ChatOps了。而不管你的这个Chat是在聊天工具里面，在网页里面，甚至是在空气里面（我指的是语音的场景），这是ChatOps的 高级境界。

(/apps/redi
utm_sourc
banner-clic

ChatOps应用场景

说了这么多，ChatOps到底能够给我们带来什么好处呢？

1. 提升答疑效率。不追求解决100%答疑工作，而可以前80%的常见问题转化为聊天机器人自动返回结果。
2. 添加聊天软件内置的GitHub/GitLab机器人，可以把代码提交，issue变更等情况同步到项目开发群里面，提升沟通效率。
3. 构建自己的业务机器人。利用开放平台提供的框架，由于它支持调用外部WebHook，因此你的聊天机器人将不仅仅用来完成简单的答疑，而可以和你的后端服务进行交互，前提是你的后端服务一定要暴露API！如监控，调度，压测，弹性伸缩机器人等等。这些机器人将取代人工操作的形式，将信息和操作流程全部透明化。
4. 国际化。在国际化的场景下，我们的Ops无需开发各种语言的Web界面，我们只需要提供API，让聊天机器人解决语言的问题，人工智能以及NLU能帮助聊天机器人更好的理解用户的意图。Let chatbot talk to your API, and people talk to your chatbot！而如果有跨时区的团队合作，团队等于有了7*24永远在线的好兄弟，随时沟通遇到的问题，同步最新状态等等。

WebHook平台

一旦ChatOps的流行，会促成业务机器人大量涌现，这些机器人和集团内部各大Ops系统进行交互，但各大系统返回的API响应，必定各不相同，如果要适配到ChatBot的标准接口上，需要各大Ops改写代码来适配。这会是一个比较繁琐的过程。比较好的办法是，通过一段简单代码来完成这个接口的适配工作。

那么问题来了，这些代码存在什么地方？在哪里执行？

这些代码的特点都是事件触发，执行完之后立即结束，生命周期很短，调用频率也不高。是Servless很好的应用场景。随着阿里云的Servless解决方案 FunctionCompute 于今年4.26正式上线，能否有一个平台能够用于托管和运行这些WebHook？

当然，Webhook也还有一些问题，比如WebHook中和各个Ops进行交互时的权限控制问题，是否需要一个地方能颁发统一的access_token？再有就是WebHook的开发，运行时的调试的问题等等。

未来

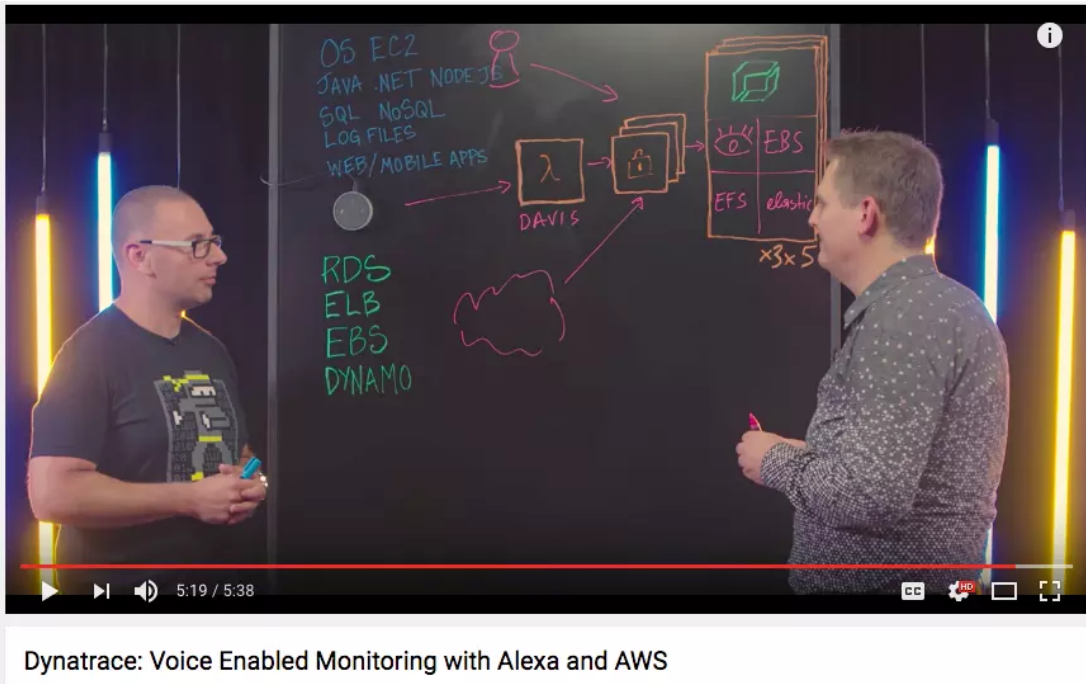
ChatOps的下一步是什么？在文字交互的基础上，语音交互 VoiceOps 正向我们走来。

首先，各大巨头的语音助手诸如Apple Siri, Amazon Alexa, Google Assistant, 都已经能通过语音和用户开发的应用进行交互了，有一天你面对的可能不是一个聊天软件，而是一个语音助手，帮助你申请权限，发布应用，确认各种状态。。。



而在监控诊断方面，APM的巨头之一Dynatrace则宣布了虚拟语音助理Davis (The Jarvis for Devops)，可以通过聊天语音的方式询问Davis，昨天晚上有没有啥问题啊？ Davis说，我发现了xxx个问题，有xxx个值得的你关注一下，balabala。。

(/apps/redi
utm_sourc
banner-lic



从这个视频截图中也能看到，Davis其实就是借助了Amazon Alexa，然后跑在Amazon Lambda上的一段程序，背后也是调用了Dynatrace的API。

感兴趣的同学可以看这个 (<https://link.jianshu.com?t=https://www.youtube.com/watch?v=ghuFQTCer7I>)和这个 (<https://link.jianshu.com?t=https://www.youtube.com/watch?v=HF5IO-nyu8Q>)视频

目前这一领域真的是非常非常新，能找到的资料很少。

总结

ChatOps在CommandOps阶段已经非常成熟，国外不少公司已经开始大规模的使用，而NLUOps则处于发展阶段，VoiceOps则基本还处于萌芽状态。
接下来我们准备在团队内试行ChatOps，看看它究竟能能否在效率上带来提升！

再往后呢？

VRops, GestureOps, MindOps, NoOps？（这里就纯属YY了哈哈）

参考资料

- <http://www.infoworld.com/article/3062703/devops/how-chatops-can-revolutionize-your-business.html> (<https://link.jianshu.com?t=http://www.infoworld.com/article/3062703/devops/how-chatops-can-revolutionize-your-business.html>)
- <https://www.atlassian.com/blog/software-teams/what-is-chatops-adoption-guide> (<https://link.jianshu.com?t=https://www.atlassian.com/blog/software-teams/what-is->



- chatops-adoption-guide)
- <https://deploybot.com/guides/chatops-deploybot-slack> (<https://link.jianshu.com?t=https://deploybot.com/guides/chatops-deploybot-slack>)
 - <http://nordicapis.com/12-frameworks-to-build-chatops-bots/> (<https://link.jianshu.com?t=http://nordicapis.com/12-frameworks-to-build-chatops-bots/>)
 - <https://olegkorol.de/2017/04/23/Creating-a-smart-ChatBot-for-Slack> (<https://link.jianshu.com?t=https://olegkorol.de/2017/04/23/Creating-a-smart-ChatBot-for-Slack>)
 - <https://developers.google.com/actions/develop/conversation> (<https://link.jianshu.com?t=https://developers.google.com/actions/develop/conversation>)
 - <https://docs.botframework.com/en-us/> (<https://link.jianshu.com?t=https://docs.botframework.com/en-us/>)
 - <http://www.cio.com/article/3172714/devops/virtual-assistants-move-into-devops.html> (<https://link.jianshu.com?t=http://www.cio.com/article/3172714/devops/virtual-assistants-move-into-devops.html>)

(/apps/redi
utm_sourc
banner-clic

小礼物走一走，来简书关注我

赞赏支持

📖 日记本 (/nb/12416175) 举报文章 © 著作权归作者所有



13太保 (/u/39c54f1a1d1c)


写了 5324 字，被 15 人关注，获得了 6 个喜欢
(/u/39c54f1a1d1c)

+ 关注

喜欢 | 6



更多分享



下载简书 App ▶

随时随地发现和创作内容



(/apps/redirect?utm_source=note-bottom-click)



登录后发表评论

登录 (/sign-in?utm_source=desktop&utm_medium=not-signed-in-comr



1条评论 只看作者 按时间倒序 按时间正序



hcjhcy (/u/165e062e2d12)

2楼 · 2017.06.06 11:38

(/u/165e062e2d12)
不错

1人赞 回复

(/apps/redi
utm_sourc
banner-clc

被以下专题收入，发现更多相似内容

视野 (/c/29a3160d6d65?utm_source=desktop&utm_medium=notes-
included-collection)

(/p/92e46ab43b68?)

utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendatio
零到十亿美元的商业传奇是怎样打造的 (/p/92e46ab43b68?utm_campaign=...想必大家都想知道Slack赢得用户的青睐以及获得这样令人羡慕的成就的原因是什么。因为对于Slack来说花
几百万美元购买广告牌、参与大型的营销竞争、煞费苦心地制定电子邮件战略、公司中有 CMO，这些都是...

zolso (/u/22813799945f?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

iOS开发 非常全的三方库、插件、大牛博客等等 (/p/63f47b492b5a?utm_c...

转自：http://www.jianshu.com/p/247a75c0fdd8 UI 下拉刷新 EGOTableViewPullRefresh- 最早的下拉刷新控
件。 SVPullToRefresh- 下拉刷新控件。 MJRefresh- 仅需一行代码就可以为UITabl...

yunxiu (/u/0e04231643ce?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

Amazon Simple Storage Service (S3) 常见问题 (/p/ae0e8752fa81?utm_...

https://aws.amazon.com/cn/s3/faqs/#sia_anchor Amazon Simple Storage Service (S3) 常见问题 一般性问
题 区域 计费 安全性 数据保护 Amazon S3 Standard – Infrequen...

守望者_1065 (/u/b3786e48e963?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

网易云Web SDK 开发手册 (/p/14bc4d1a642b?utm_campaign=maleskin...

点击查看原文 Web SDK 开发手册 SDK 概述 网易云信 SDK 为 Web 应用提供一个完善的 IM 系统开发框架，
屏蔽掉 IM 系统的复杂的细节，对外提供较为简洁的 API 接口，方便第三方应用快速集成 IM 功能。网易云信...

layjoy (/u/1bc60c77e4ec?)


utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio



不算总结的一条全年心得 (/p/22a4cd1aa3f1?utm_campaign=maleskine&...

2016年接近尾声，我的2016年也即将结束。如果说高考钱和高考后是两个方向，那么，今年是我明白许多事

的一个分水岭。我没有计算今年发生了什么事，但就是要比往年的每一年都要多。记得上一次事多的时候...


 曹曹曹曹小妹 (/u/e470e0e2248a?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio (/apps/redi
utm_sourc
banner-clip
(/p/53705407cab3?)



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendatio
我愿做一只面朝大海春暖花开的喵 (/p/53705407cab3?utm_campaign=ma...


世界很难 冬天很冷 爱很暖。前路未卜 还有梦。有时候最简单的游戏，我也可以玩一天，和我一样的请举手
~ 偶尔也要给自己一些时间，发发呆上上网，谁说慢慢生活就是浪费呢？冬天的时候，只想蜷在暖暖的毯...

 小日子APP (/u/781ea2cdd033?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

万字挑战7：欲速则不达 (/p/6e55a8d3f3ef?utm_campaign=maleskine&ut...

前段时间一直发现自己做任何事情都很匆忙，每次都像赶鸭子上架一样，匆匆忙忙。早上一起来，赶着刷牙，赶着忙这忙那，感觉自己没法慢下来，一慢下来就焦虑不安。记得十月份国庆回家的时候，家人给我...

 石韧 (/u/4a8cc5fe4538?)


utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

(/p/36b9690ec475?)



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendatio
不了解自己，难道真想一辈子试错？ (/p/36b9690ec475?utm_campaign=...


很多人都出奇地固执。比如看书，明明科技类的文献看不明白非要死磕自己，翻了一上午啥都没记住只剩下焦躁；比如组装工具，明明自己不擅长非要摆弄挑战极限，最后看着散落一地的零件真想踢一脚；比如明...

 鹤飞飞飞飞飞 (/u/942b4f813146?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

洋油灯 (/p/bbd809db95c7?utm_campaign=maleskine&utm_content=no...

小时候，故乡汝阳小城用电不普及，山村里就更难有电；只有政府机关、影剧院、厂矿、生产队的钢磨抽水机等才用得上电。一到晚上，家家户户点起洋油灯，红黄色的光便点亮每一扇窗户。那灯虽不如电灯明亮...

 马建国诗文 (/u/dec14297e852?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

