The Dennis Botman Story

A Tale of Next-Level Chatops

A Technical Paper prepared for SCTE by

Michael Winslow
Senior Director, Software Development and Engineering
Comcast Corporation
michael_winslow@comcast.com

Ryan Emerle
Senior Principal Engineer
Comcast Corporation
ryan_emerle@comcast.com

Mia Kuang
Software Engineer
Comcast Corporation
mia_kuang@comcast.com

Table of Contents

Title **Page Number**

1. Introduction ... 3
   1.1. What is Chatops? ... 3
   1.2. Why do we need Chatops? ... 3
2. Problem Statement ... 4
   2.1. Small Team Supporting a Large Organization ... 4
   2.2. Customers Do Not Read Documentation ... 4
   2.3. Providing Real-Time Support ... 5
   2.4. Providing After-Hours Support ... 5
   2.5. Is Automation the Answer? ... 5
3. Getting Started Getting Started ... 7
   3.1. To Build or to Buy ... 7
   3.2. The “Big Ideas” Phase ... 7
   3.3. Perfect is the Enemy of Good ... 8
4. Dennis Botman is Born / How it works ... 8
   4.1. Initial Feature Set ... 8
   4.2. Basic Interaction (FAQs) ... 10
   4.3. Pre-Processing Text/Questions From Users ... 11
5. Increasing the Usage of the Bot ... 12
   5.1. Roadshows and Reminders ... 12
   5.2. One-on-One Conversations with the Bot ... 12
   5.3. Using the Bot as a Proxy ... 13
   5.4. Letting the Bot be the Bad Guy (Unsolicited Responses) ... 13
   5.5. Keeping Our Engineers Involved ... 14
6. Observing the Lasting Effects ... 15
   6.1. Overview of the Data ... 15
   6.2. Re-calculating ROI based on real numbers ... 15
7. Conclusion ... 16

Abbreviations ... 17

Bibliography & References ... 17

List of Figures

Title **Page Number**

Figure 1 - Typical Chatops flow for Dennis Botman ... 3
Figure 2 - Vinyldns-bot high level design ... 9
Figure 3 - Basic Dennis Botman conversation ... 10
Figure 4 - Creating a dialog between Dennis Botman and a third party ... 13
Figure 5 - An example of the “lookup” command ... 14
Figure 6 - VinylDNS Support Channel Interactions ... 15

List of Tables

Title **Page Number**

Table 1 - Estimated cost for a chatbot paid solution ... 7

1. Introduction

What do you do when you have an amazing team and you want to make them even better? The 1995 Chicago Bulls added Dennis Rodman. Well, our team at Comcast created Dennis BOT-man: An automated Chatops bot that helped us re-imagine how we support our internal engineering teams.

1.1. What is Chatops?

ChatOps is a collaboration model that connects people, tools, process, and automation into a transparent workflow (see Figure 1). With our Chatbot, Dennis Botman, we've taken an iterative approach to add functionality over time. As we will get into later in the paper, our primary objective is to connect people with knowledge resources like frequently asked questions FAQs and how-to documentation.

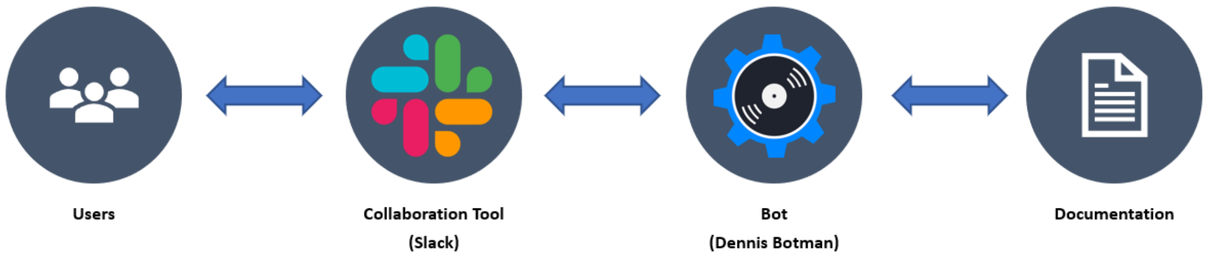


Figure 1 - Typical Chatops flow for Dennis Botman

1.2. Why do we need Chatops?

To understand why we needed this level of automation, it helps to know the scale of the application we support.

Comcast's open-source domain name system (DNS) management tool, VinylDNS, allows millions of DNS entries to be controlled by hundreds of individual groups throughout the company (Cleary, 2020). This decentralized governance model is a marvel on its own. One software developer, Stephanie Hingtgen, described her team's usage of VinylDNS in the following way:

> *"We leverage the VinylDNS system to instantly provision A and AAAA records for all new VIPs in our zone, as well as giving the user the ability to provision CNAME records. This was crucial for rolling out IPv6 VIPs in RDEI (Rapidly Deployed Elastic Infrastructure) so that users would have an easy way of remembering their VIP name."*

To ensure the reliability of this critical application, our group adopted a DevOps[1] model where our software engineers rotated to support the product. The primary means by which to engage our support team was by asking questions in our Slack support channel.

Before long, our support channel was filled with questions from our users; many of which were already answered in our support documentation. Valuable time that our engineers could have been spending developing new software and features were instead spent repeatedly answering the same basic questions. Something had to be done.

[1] DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity

In this paper, we'll tell the story of how we coded and delivered Dennis Botman to be a true member of the support team. We'll explain how Dennis Botman was able to answer over 35% of the questions being asked in our support channel. We will show how adding powerful tooling within the bot allowed engineers all over the company to solve DNS issues instantly. And perhaps most fascinating, we will use metrics to demonstrate how our support bot empowered our users by arming them with a self-service mindset.

After reading this paper, if your team would like to take advantage of the features we developed in Dennis Botman, feel free to visit our open-source project on Github: https://github.com/vinyldns/vinyldns-bot

2. Problem Statement

2.1. Small Team Supporting a Large Organization

The world of DNS is massive. Everything that is on the internet, from servers to automated teller machines (ATM) machines to your smartphone, are all addressable by internet protocol (IP) addresses which are often mapped to DNS records.

When tasked with managing millions of these records, which service hundreds of teams throughout Comcast, we set out to build a service to enable users to manage their own records. Born out of the necessity to reduce time-to-market, VinylDNS is the user-facing service we created that enables self-service for DNS.

What happens when thousands of users across hundreds of organizations suddenly have access to DNS records and the inherent complexity therein? They're going to have questions and they're going to need answers.

Suddenly, what was once a utopian dream of a self-sufficient, self-service, tool becomes a conduit to an influx of support requests. This deluge of questions simply shifted the burden and complexity of DNS management from the DNS engineering team to the VinylDNS service team.

Quickly we were scrambling to extricate ourselves from the critical path of thousands of users. To do so, we needed to find ways to offload our support personnel while still getting answers to the users and getting DNS records provisioned.

2.2. Customers Do Not Read Documentation

One of the quickest and easiest ways to get answers for users was to thoroughly document the service. With VinylDNS, the service and its corresponding APIs and tooling were all thoroughly documented before even being released to users.

Documentation of every aspect of the system was written. Reams of documentation detailed every nuance of the service and its corresponding user interface. Months of writing and revising lead to a thorough reference of the system and its behavior.

The problem? Users unfortunately do not tend to read documentation. This isn't because users are lazy, or unwilling; they simply have more important problems to solve. As authors of a user-facing service, we cannot assume that we are the center of our users' universe. Our service is likely a small part of the overall problem being solved, and as such we cannot expect thousands of users to take several hours to pore over reams of documentation.

2.3. Providing Real-Time Support

In order to overcome this tendency to ignore the documentation, we needed to provide a means by which users could be directed to the subset of documentation that addresses their questions.

A logical next step was to provide real-time support. This entailed creating a channel in Slack where users could come and ask questions. The channel would be staffed by VinylDNS engineers who would help users get to their answers.

Upon establishing real-time support, the support personnel were inundated with questions. Among these were a set of questions that kept appearing. As such, the natural next step was to distill some of the documentation down into FAQs. The FAQs provided more direct answers and saved the users and support personnel time sifting through documentation.

While creating FAQs helped to reduce the influx of support questions, a pattern started to emerge. Users would come to the support channel, ask a question, and then a support person would find the relevant FAQ, copy the link, and paste it into the support channel as a response.

During this period, there were three to four engineers spending upwards of thirty percent of their time answering questions, searching the documentation on behalf of users, and pasting FAQ links into the support channel.

2.4. Providing After-Hours Support

Our team monitored our Slack support channel for one week and discovered that 13% of the questions from our customers happened after hours. There is no guarantee that a chatbot would solve their problems, but our customers would have more options when our engineers are not available. A simple reminder of our on-call hours is better than no response at all.

2.5. Is Automation the Answer?

It soon became clear that the repetitive nature of the real-time support that was being offered was untenable. We had engineers whose time is better spent improving the product than acting as glorified search engines.

When faced with repetitive tasks, there is a tendency toward automation. As engineers, we evaluated the landscape and began to consider how we might automate the repetitive task of searching documentation, copying links, and pasting them into the support channel.

The cost of automation is not free, however, so we needed to estimate the return-on-investment (ROI) for the automation.

The general formula for ROI *(Fernando, 2021)* is:

$$ROI = \frac{Net\ Value\ of\ Investment}{Cost\ of\ Investment} \times 100\%$$

In the case of automation, our value and cost parameters are in units of time. As such, we can restructure the equation as follows:

$$ROI = \frac{Time\ Saved - Time\ Spent}{Time\ Spent} \times 100\%$$

This simple formula can be a bit deceiving, as Time Spent needs to include the initial time to build the automation plus all future time spent maintaining it. Similarly, Time Saved is also cumulative. Further, determining, or predicting, Time Saved can be challenging. In this case, we're focused on the estimated time saved by the predicted reduction in time spent on support duties. Time saved by reduction in human error, training personnel and other factors are not being considered.

For the purposes of calculating a value, we can limit the ROI projections and calculation to a discrete time period. For the following example, we limit the calculation to one year.

Assume we have three engineers spending 30% of their time supporting users. Given a 40-hour work week, we can calculate the support cost, in time, for a single week:

$$Support\ Cost_{week} = 3\ engineers \times (40\ hours\ \times 30\%)$$
$$Support\ Cost_{week} = 3\ engineers \times\ 12\ hours$$
$$Support\ Cost_{week} = 36\ hours$$

Let's assume that we can implement some automation in four weeks at 40 hours per week.

$$Automation\ Cost_{total} = 4\ weeks\ \times 40\ hours$$
$$Automation\ Cost_{total} = 160\ hours$$

Let's further assume that our analysis indicates that the automation is projected to save 25% of our support personnel's total time.

$$Automation\ Savings_{week} = Support\ Cost_{week} \times 25\%$$
$$Automation\ Savings_{week} = 36\ hours \times 25\%$$
$$Automation\ Savings_{week} = 9\ hours$$

If our cost of automation is a total of 160 hours, then we can calculate the ROI for a year:

$$ROI_{year} = \frac{(Automation\ Savings_{week} \times 52\ weeks) - Automation\ Cost_{total}}{Automation\ Cost_{total}} \times 100\%$$

$$ROI_{year} = \frac{(9\ hours\ \times 52\ weeks) - 160\ hours}{160\ hours} \times 100\%$$

$$ROI_{year} = \frac{468\ hours - 160\ hours}{160\ hours} \times 100\%$$

$$ROI_{year} = \frac{308\ hours}{160\ hours} \times 100\%$$

$ROI_{year} \approx 192\%$

The resulting calculation shows that if we invest 160 hours in automation to recover 25% of the time spent performing support activities, we'll see a projected return of 192%. For every hour spent automating, 192% of that time is returned as time saved over a one-year period.

3. Getting Started Getting Started

3.1. To Build or to Buy

With an estimated ROI of 192%, we were confident that we were going to take steps toward automation. But before deciding to invest our engineering time to creating a custom chatbot for our internal customer base, we did some initial research on the costs of purchasing a solution. We found that adopting a chatbot service can possibly get quite costly.

According to mobilemonkey.com (MobileMonkey, 2021), once you factor in the cost of the platform along with the salary related costs for setup, development, and maintenance, you could be paying over $60k in the first year alone.

Table 1 - Estimated cost for a chatbot paid solution

	In-House Chatbot Costs	Agency Chatbot Fees
Chatbot Software Platform	$50-$500/month	$50-$500/month
Chatbot Setup and Development	Salaries (5-100 hours of work)	$500-$2,500
Ongoing chatbot support and maintenance	Salaries (0-10 hours of work per week)	$50-$5000/month

We decided it would be better to develop our own bot using opensource solutions and allowing our engineers to solve our specific problems. We chose to start with the Github's Hubot Framework (Metz, 2015) which is written in CoffeeScript and node.js. Hubot handles basic chat communication, which saved us a lot of time. But we required much more intelligence built into our bot if it was going to be useful. We would need to extend the functionality of Hubot, but by how much?

3.2. The "Big Ideas" Phase

During our initial brainstorming sessions, one thing that the team insisted was that the bot should be conversational rather than a glorified command line interface (CLI). This aspect did cause some team members to really think BIG:

- "We need to learn natural language processing / understanding (NLP/NLU)."
- "We cannot do this without artificial intelligence."
- "Machine learning is essential."

After several meetings that appeared to get us no closer to a starting point, the team began to lose interest. It was starting to feel like this task was simply too large an undertaking. As discussed in section 2.5 (Is Automation the Answer?), there just did not seem to be enough value to justify this level of effort.

Our problem was that we were trying to design the perfect system right from the planning phase. The sheer size and complexity of the solution we were proposing was putting us in a state of analysis paralysis. We would either need to narrow our scope or scrap the idea all-together.

3.3. Perfect is the Enemy of Good

Anyone who has ever read Stephen R. Covey's best-selling book **7** Habits of Highly Effective People (Covey, 1989) will tell you that to "begin with the end in mind" is critical. But be careful not to misconstrue this message with the idea that everything must be known and understood before beginning work on a software project. There are so many benefits to be had by taking an iterative approach.

In a 2019 article published by Forbes on the topic of Machine Learning projects, they state that "87% of projects do not get past the experiment phase and so never make it into production" (Dans, 2019). This is because software development teams far too often attempt to "boil the ocean" rather than identifying bite-sized chunks of functionality to deliver fast. It is better to have a product mindset and get users to begin interacting with your software in order to give valuable feedback.

If we wanted to be a part of the 13% of projects that actually see production, we were going to have to focus on the core problem we are trying to solve. We decided to clearly state our purpose and simplify our initial feature set. Once we were able to agree on a small set of essential requirements, Dennis Botman was ready to come to life!

4. Dennis Botman is Born / How it works

4.1. Initial Feature Set

We decided on the following technical requirements:

- The bot will be available in 0..n Slack support channels
- The bot can accept free text "interactions" as well as exact-match "commands"
- The bot can respond with answers to FAQs
- The bot can respond with links to support documentation and real-time dashboards
- Users can have direct message (DM) conversations with the bot
- The bot MUST log all interactions with the bot (to improve support)

A high-level design of our implementation is shown in Figure 2.

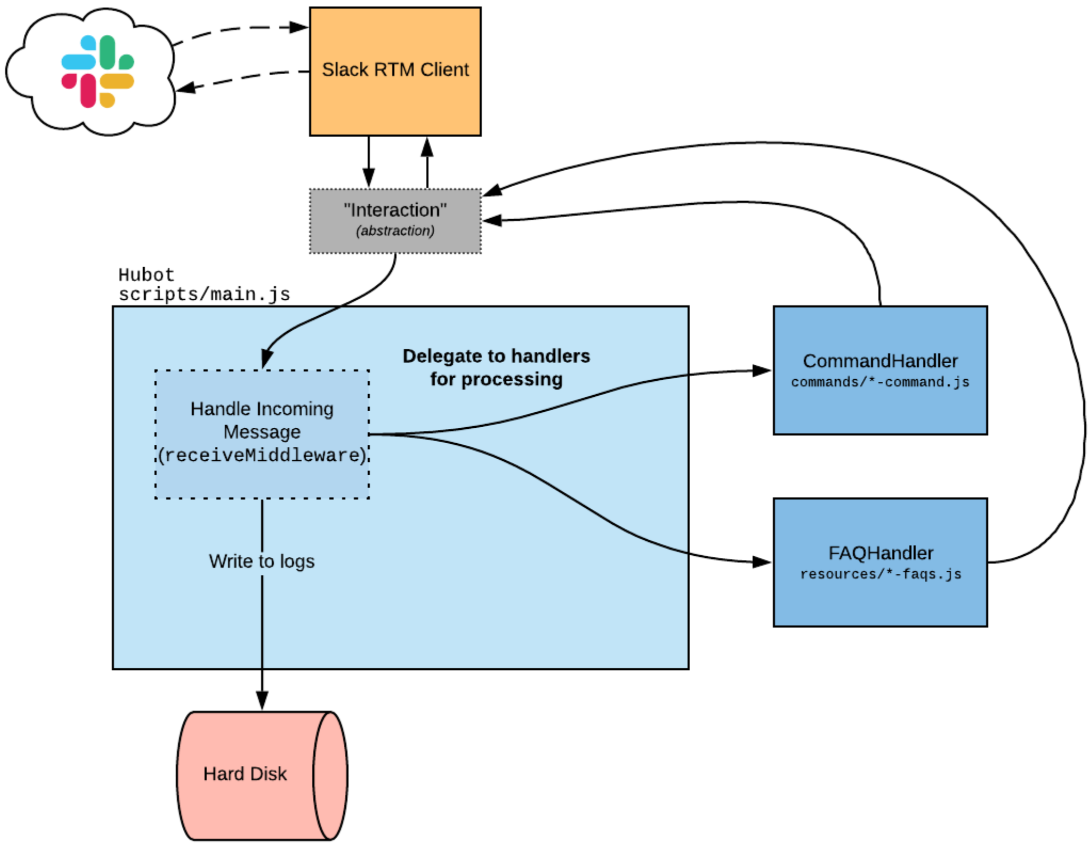
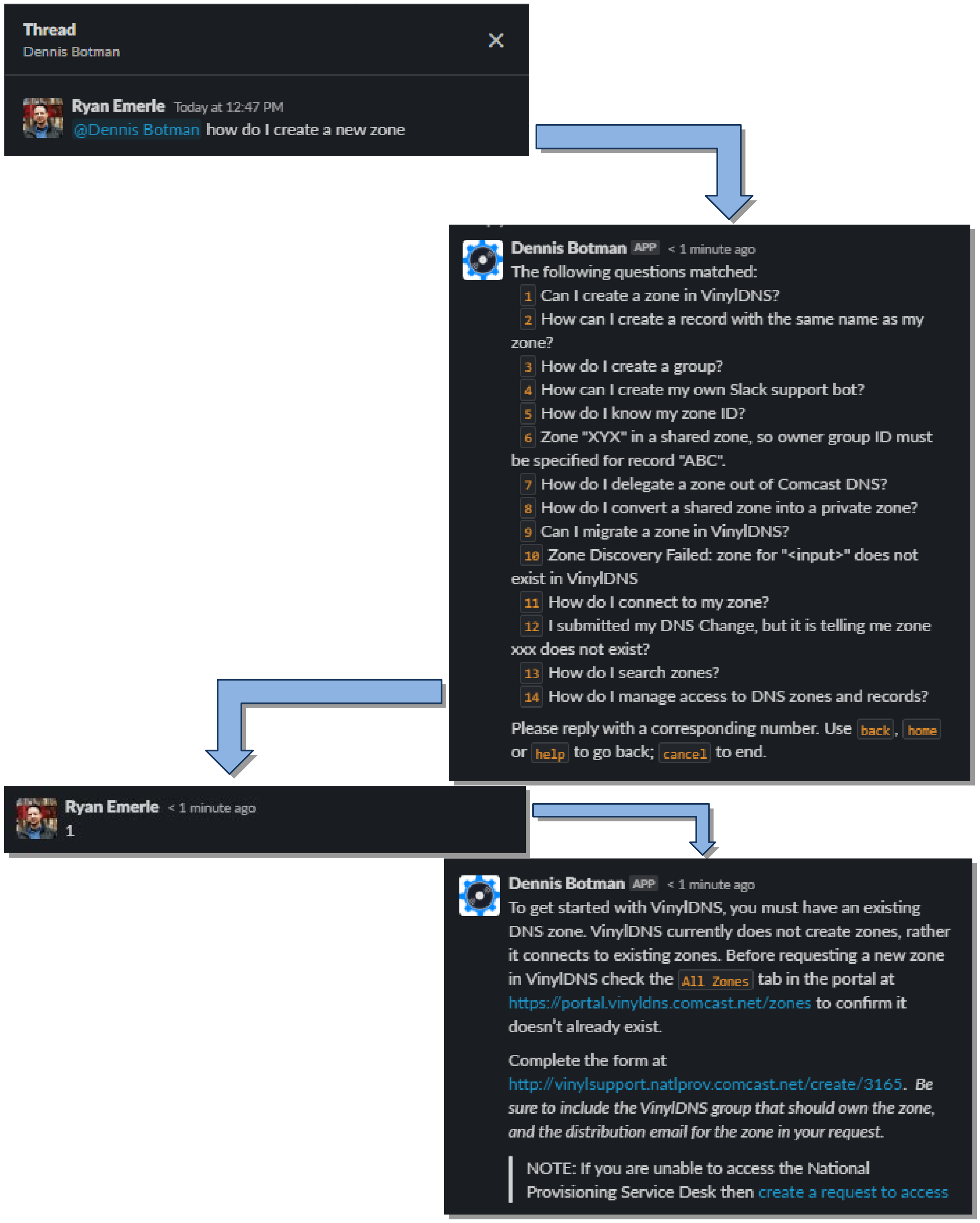


Figure 2 - Vinyldns-bot high level design

4.2. Basic Interaction (FAQs)

Figure 3 shows a common example of a user interacting with Dennis Botman.


Thread
Dennis Botman

Ryan Emerle Today at 12:47 PM
@Dennis Botman how do I create a new zone

Dennis Botman APP < 1 minute ago
The following questions matched:
1 Can I create a zone in VinylDNS?
2 How can I create a record with the same name as my zone?
3 How do I create a group?
4 How can I create my own Slack support bot?
5 How do I know my zone ID?
6 Zone "XYX" in a shared zone, so owner group ID must be specified for record "ABC".
7 How do I delegate a zone out of Comcast DNS?
8 How do I convert a shared zone into a private zone?
9 Can I migrate a zone in VinylDNS?
10 Zone Discovery Failed: zone for "<input>" does not exist in VinylDNS
11 How do I connect to my zone?
12 I submitted my DNS Change, but it is telling me zone xxx does not exist?
13 How do I search zones?
14 How do I manage access to DNS zones and records?

Please reply with a corresponding number. Use `back`, `home` or `help` to go back; `cancel` to end.

Ryan Emerle < 1 minute ago
1

Dennis Botman APP < 1 minute ago
To get started with VinylDNS, you must have an existing DNS zone. VinylDNS currently does not create zones, rather it connects to existing zones. Before requesting a new zone in VinylDNS check the `All Zones` tab in the portal at https://portal.vinyldns.comcast.net/zones to confirm it doesn't already exist.

Complete the form at http://vinylsupport.natlprov.comcast.net/create/3165. *Be sure to include the VinylDNS group that should own the zone, and the distribution email for the zone in your request.*

> NOTE: If you are unable to access the National Provisioning Service Desk then create a request to access

Figure 3 - Basic Dennis Botman conversation

4.3. Pre-Processing Text/Questions From Users

Users can interact with Dennis Botman by asking any question via direct messaging or in a Slack channel. Dennis Botman can process a user's question by searching through a set of FAQs files and return a subset of matched questions.

While the Hubot Framework handles the directing of traffic between the user and the bot, we still had to program the logic in determining how the incoming questions were processed. To do this, we utilized some common techniques around "Text Cleaning" and "Pre-Processing."

The bot pre-processes a user's input and extracts meaningful words. Then it performs the search for matching words from all the FAQs in the repository. Any FAQ that has one word or more matched is a candidate. Each FAQ is presented as a question form itself, with hidden extra terms that can be used to match with user input, and exclusion terms to be removed from the search. Depending on the number of matched questions, the number of matched meaningful words in each question, and the configured limit for number of results, the search will return a list of FAQs in a specific ranked order. If there is only a single match, the response to that FAQ will be provided directly. If there are multiple possible matches, the user will be prompted to choose from them using an enumerated response.

Using the example question from above, let's step through how Dennis Botman interprets the input:

Starting String:

"How do I create a new zone?"

Step 1: Convert text to lowercase

Converting all the incoming request text to lowercase allows us to do exact string comparisons against our library of FAQs given that we also convert the library to lowercase strings.

- result: *"how do i create a new zone?"*

Step 2: Remove unwanted characters

Special characters and punctuation can cause our results to be inaccurate. It helps to scrub the incoming text of any characters that may be troublesome.

- result: *"how do i create a new zone"*

Step 3: Remove stop words

Stop words are words which are filtered out before or after processing of natural language data (text). Words like "a", "do", and "is" are considered "stop words." We maintain our own list of stop words to increase our chances of getting a high-confidence result to our text search.

- result: *"create new zone"*

Step 4: Tokenization

Tokenization (in lexical analysis) is the process of demarcating and possibly classifying sections of a string of input characters. In other words, we separate each word in our string so that they may be used individually. This will help create the word match count needed for the final step.

- result: [*“create”, “new”, “zone”*]

Step 5: Ranking

By counting the number of matches that we have of our tokenized words within our library of FAQs, we can rank the confidence of our results. You can clearly see why the most relevant results appear highest in our list.

- “Can I **create** a **zone** in vinyldns?” – (2 matches)
- “How can I **create** a record with the same name as my **zone**?” – (2 matches)
- “How do I **create** a group?” – (1 match)

5. Increasing the Usage of the Bot

5.1. Roadshows and Reminders

In order to realize the benefit of our investment of time into creating a bot to automate support of the VinylDNS service, we need our users to engage with the bot. Not only must users engage with the bot, but they must also be able to get the answers they’re looking for, so that they do not require human intervention.

The first step in increasing engagement is a logical one – tell the users that the bot exists. After initial development of the bot, the VinylDNS team found opportunities to speak at internal conferences in order to raise awareness across organizations. Through technical conferences and demonstrations, we were able to promote the existence of the bot and its many benefits.

While we evangelized the bot across multiple organizations, we also needed to bring attention to the bot where support was happening – in our Slack channel. To do so, we tackled the low-hanging fruit first. We simply added details about the bot in the channel topic. The channel topic is the place where users can find information about the channel and the various resources offered.

While include the bot in the channel topic was essential, it did not draw immediate attention to the bot. So, in order to supplement this, we added a daily “message of the day.” This is a message that is posted to the support channel every morning. The message is posted by the bot itself and it details the existence of the bot, how to use it, and other details about the support channel.

With the addition of the “message of the day” we saw a marked increase in bot interaction. However, on occasion, the message of the day would be scrolled out of view by support discussions. As such, we also added a notification for all users who first join the support channel. Upon joining, new users receive a message detailing the existence of the bot and how to interact with it to get answers.

5.2. One-on-One Conversations with the Bot

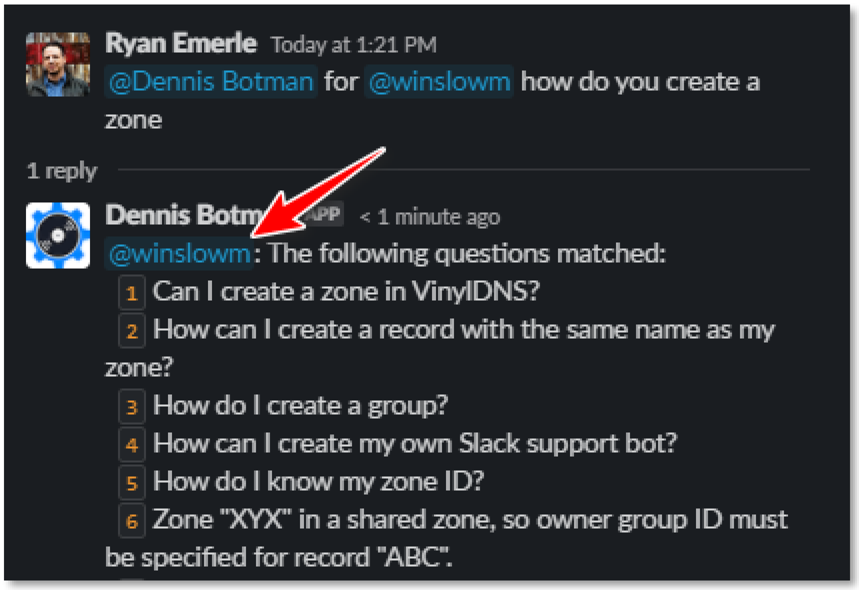
After making a concerted effort to promote the existence of the bot, we needed to find ways to encourage users to interact with it.

First, we wanted to reduce the anxiety of users experimenting with a bot in a Slack channel with over 1,000 users. As such, one of the first features we added was the ability to DM the bot. This allowed users to interact with the bot in a private channel, independent of the main support channel. All the functionality remains the same; the only difference is that the experimentation is not broadcast to the wider community.

5.3. Using the Bot as a Proxy

As Albert Einstein once said, "[E]xample isn't another way to teach, it is the only way to teach." With that in mind, we wanted to encourage usage of the bot by interacting with it ourselves. To do this, instead of copy-and-pasting links to FAQs on our documentation website, we issued commands to the bot on behalf of the user.

To further extend this concept, we added a special command: "for <user>". This allows support personnel to issue commands on behalf of another user but establish a dialog between the bot and that user (see Figure 4).



Ryan Emerle Today at 1:21 PM
@Dennis Botman for @winslowm how do you create a zone

1 reply

Dennis Botman APP < 1 minute ago
@winslowm: The following questions matched:
1 Can I create a zone in VinylDNS?
2 How can I create a record with the same name as my zone?
3 How do I create a group?
4 How can I create my own Slack support bot?
5 How do I know my zone ID?
6 Zone "XYX" in a shared zone, so owner group ID must be specified for record "ABC".

Figure 4 - Creating a dialog between Dennis Botman and a third party

5.4. Letting the Bot be the Bad Guy (Unsolicited Responses)

This feature was the result of collaboration with engineers in other groups, namely the groups that manage the DNS servers. One issue that they identified is that users would often create a support ticket for some DNS change, then immediately go to the real-time support channel and ask about the status.

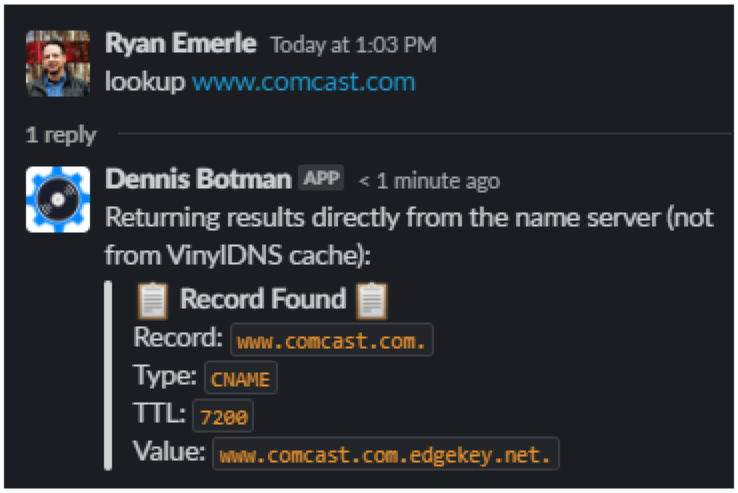
In order to reduce the need for human intervention when a ticket was already created, we introduced the concept of "unsolicited messages". Unsolicited messages are messages sent by the bot that are not the result of a user interaction with the bot. As an example, a user might come to the support channel and say, "I created a ticket XYZ-123, can someone take a look?" The bot uses pattern matching to detect the presence of a ticket number, and automatically responds to the requestor with a message letting them know that the ticket will be addressed in the order in which it was received and no further action is required on their part.

5.5. Keeping Our Engineers Involved

Once user engagement began to increase, we found additional opportunities to add functionality. Instead of simply providing answers to questions, the bot could also retrieve data from external sources in order to answers questions.

Keeping engineers involved in the automation of support activities was the key to making the bot more useful and, as a result, better equipped to resolve user questions without human intervention.

An example is the addition of a "lookup" command. This command queries the DNS backend servers to return record resolution details back to the user (see Figure 5). This is in lieu of asking the user to use cryptic command line utilities to query DNS resolvers. In the case of "lookup," the user could simply message the bot "lookup <FQDN>" (where FQDN is a fully qualified domain name; e.g., "lookup www.comcast.com"). The result is a human-readable DNS lookup.


Ryan Emerle Today at 1:03 PM
lookup www.comcast.com

1 reply

Dennis Botman APP < 1 minute ago
Returning results directly from the name server (not from VinylDNS cache):

📋 Record Found 📋
Record: `www.comcast.com.`
Type: `CNAME`
TTL: `7200`
Value: `www.comcast.com.edgekey.net.`

Figure 5 - An example of the "lookup" command

Similar types of commands were added that increased the utility of the bot and reduced the prerequisites required of users trying to interact the DNS services.

Engagement with a wider community proved to be invaluable. Through collaboration with engineers across multiple organizations we were able to significantly increase the ability for users to get to answers more quickly.

Seeing the power of collaboration in action, and given that VinylDNS is an open-source project, we open-sourced the VinylDNS bot code as well. Through greater exposure we hope to evolve the project to meet the needs of larger and more diverse audiences.

6. Observing the Lasting Effects

6.1. Overview of the Data

Determining the impact of our conversational chatbot would require us to log every interaction. This enables us to compare the number of times our internal customers were able to get the answers they needed without requiring human involvement. In addition to being able to put these results into buckets of human versus bot support, we were also able to determine exactly how our customer prefer to interact with the bot (see Figure 6).

The following results were over a 30-day period:

- 711 total interactions
- 347 interactions with the bot
- 260 of those were direct messages
- 7 of those were unsolicited

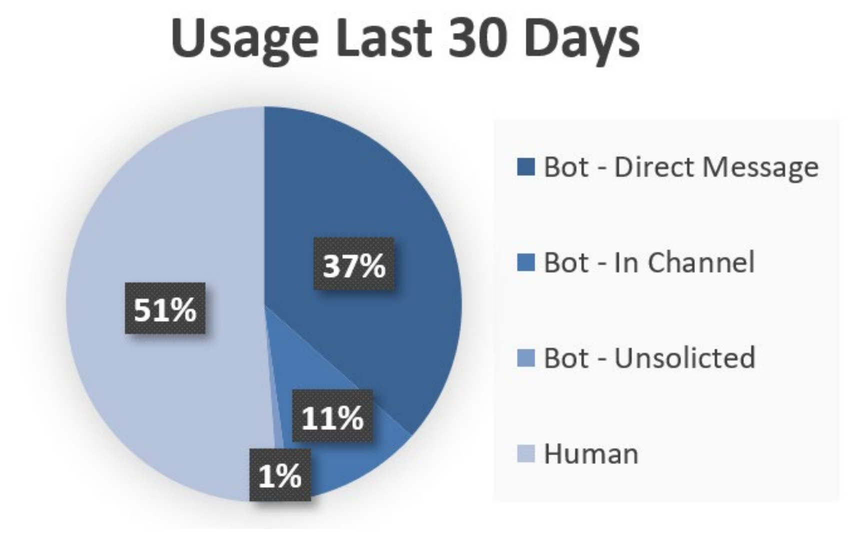


Figure 6 - VinylDNS Support Channel Interactions

According to our findings, total bot interaction was at 49% as opposed to 51% interactions with humans. This number alone does not tell us if our users were able to get to the answers they were looking for with each bot interaction. So we did a little more research.

By reviewing the users and time stamps associated with the 347 interactions with the bot, we investigated how many times those users subsequently returned to the support channel to get answers from a human. We found that 271 (78%) of those interactions required no human follow-up.

6.2. Re-calculating ROI based on real numbers

Automation Cost: Our initial time spent to develop Dennis Botman was approximately three weeks of one full-time engineer and one part-time engineer. It should be noted that we do spend small amounts of time on a continual basis to maintain the bot, but our initial time invested was less than a month.

1.5 engineers x 40 hours x 3 weeks = 180 hours of initial Automation Cost

Automation Saving: Using the statistics we gathered from the logs, we can replace our assumed time savings (25%) with the actual time savings we've observed from actual interactions with Dennis Botman.

271 bot handled requests (no human) / 711 total interactions = 38% Automation Saving

If our Support Cost remains constant at 36 hours and we multiply that by 38% instead of 25%, we get the following:

$$Automation\ Savings_{week} = 36\ hours \times 38\%$$
$$Automation\ Savings_{week} = 14\ hours$$

If our cost of automation is a total of 180 hours, then we can calculate the ROI for a year:

$$ROI_{year} = \frac{(Automation\ Savings_{week} \times 52\ weeks) - Automation\ Cost_{total}}{Automation\ Cost_{total}} \times 100\%$$

$$ROI_{year} = \frac{(14\ hours\ \times 52\ weeks) - 180\ hours}{180\ hours} \times 100\%$$

$$ROI_{year} = \frac{728\ hours - 180\ hours}{180\ hours} \times 100\%$$

$$ROI_{year} = \frac{548\ hours}{180\ hours} \times 100\%$$

$$ROI_{year} \approx 304\%$$

7. Conclusion

Software engineers solve some of the most complex problems for our customers every day to improve their product experience. Far too often, we do not use those same skills to solve our own problems to make our engineering experience more enjoyable and productive. We simply accept that tasks like on-call support, status reporting, deployments, and testing are things that have to be done manually in addition to our development tasks.

With Dennis Botman, we did more than simply play around with a Chatbot as a side-project. We, in essence, created another teammate. A teammate that could make all of our jobs easier, but only after we spent time training it, as well as making our customers comfortable with getting help from a bot. In the end, we were able to free up more time to tackle larger, more interesting problems.

Tasks to improve engineering efficiency can have a huge impact on how you grow and operate your engineering teams. Setting aside time for your engineers to solve their own problems should always be looked upon as an investment of time or resources that carries many beneficial outcomes.

Abbreviations

API	application programming interface
ATM	automated teller machine
CLI	command line interface
DNS	domain name system
DM	direct message
FAQ	frequently asked question
FQDN	fully qualified domain name
IP	internet protocol
RDEI	rapidly deployed elastic infrastructure
ROI	return on investment
NLP/NLU	natural language processing / understanding
NLU	natural language understanding
VIP	virtual ip (internet protocol)

Bibliography & References

Cleary, P. (2020, August 28). *Why Comcast open sourced its DNS management tool*. Retrieved from opensource.com: https://opensource.com/article/20/9/open-source-dns

Covey, S. R. (1989, August 15). *The 7 Habits of Highly Effective People*. Retrieved from franklincovey.com: https://www.franklincovey.com/the-7-habits/

Dans, E. (2019, July 21). *Stop Experimenting With Machine Learning And Start Actually Using It*. Retrieved from Forbes.com: https://www.forbes.com/sites/enriquedans/2019/07/21/stop-experimenting-with-machine-learning-and-start-actually-usingit/

Fernando, J. (2021, April 8). *Financial Ratios > Return on Investment (ROI)*. Retrieved from Investopedia: https://www.investopedia.com/terms/r/returnoninvestment.asp

Metz, C. (2015, October 23). *The Most Important Startup's Hardest Worker Isn't a Person*. Retrieved from WIRED: https://www.wired.com/2015/10/the-most-important-startups-hardest-worker-isnt-a-person/

MobileMonkey. (2021, August). *Chatbot Pricing: Everything You Need to Know About Chatbot Prices for SMBs*. Retrieved from https://mobilemonkey.com/: https://mobilemonkey.com/blog/chatbot-pricing/