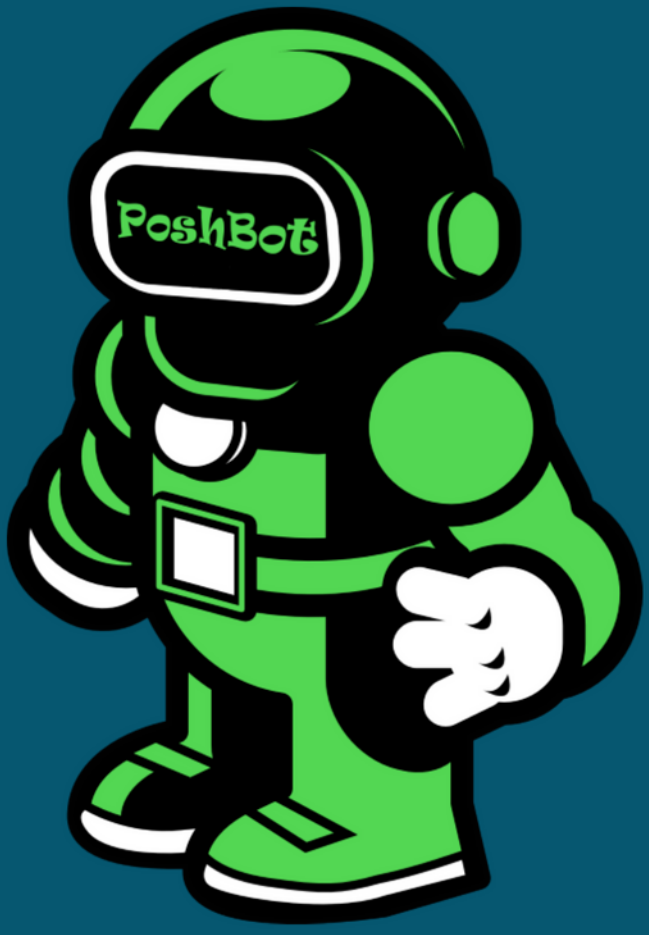
FROM THE CREATOR OF POSHBOT



CHATOPS THE EASY WAY

POWERSHELL AUTOMATION THROUGH CHAT USING POSHBOT

BRANDON OLIN

ChatOps the Easy Way

PowerShell Automation Through Chat Using PoshBot

Brandon Olin

This book is for sale at http://leanpub.com/chatops-the-easy-way

This version was published on 2019-08-09

Tweet This Book!

Please help Brandon Olin by spreading the word about this book on Twitter!

The suggested tweet for this book is:

I just purchased #ChatOpsTheEasyWay by @devblackops on @leanpub! https://leanpub.com/chatops-the-easy-way Get your #PowerShell #ChatOps on!

The suggested hashtag for this book is #ChatOpsTheEasyWay.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#ChatOpsTheEasyWay

Contents

Preface

Who is This Book For?

This book is for anyone interested in bringing ChatOps into their organization, or anyone already doing ChatOps, but wants to expand their knowledge about using PowerShell automation in their existing ChatOps workflow.

The majority of this book is technical in nature and describes "how" to accomplish tasks in PoshBot. Before that can happen though, it's important to discuss the "why" of ChatOps. These chapters are to establish a common understanding for why this way of working provides so many positive benefits to an organization.

ChatOps can be considered a subset of DevOps. Ultimately, DevOps is a cultural movement with ideas and tools forming the basis of that movement. ChatOps plays a part in that by enhancing collaboration while at the same time as enabling automation.

Typographic Conventions

PoshBot is based on PowerShell so most code samples in this book will be based on that. Code samples and commands will be highlighted either inline (`Start-PoshBot`) or in a separate code block like:

```
1  $bot | Start-PoshBot
```

Every effort will be made to keep code from wrapping to a new line, but PowerShell is a verbose language, and best practice PowerShell conventions are followed in this book, so the use of aliases or shortened parameter names will not be used. Commands that will have more than three parameters will use `hashtables` and splatting instead of specifying the parameters inline.

PoshBot's configuration file is a PowerShell `hashtable` and code samples will look like the following:

```
@{
    Name              = 'PoshBot'
    LogCommandHistory = $True
    PluginRepository  = @('PSGallery')
    ...
}
```

Links to relevant resources and websites will be added inline, like this example to PoshBot[1]. You can click on these links to view them directly in the browser when reading the eBook and PDF formats of this book or follow the URLs in below in the footnote section when the reading the hard copy version.

Asides

Occasionally, asides are used in this book to highlight certain information. Some example asides are:

This is an Aside Box

It will be used to emphasize or describe something.

This is a Warning Box

Warning will be used to call out something dangerous.

This is a Tip Box

Tips will be used to point out something useful.

This is an Error Box

Errors will be used to describe something that is broken.

[1] https://github.com/poshbotio/PoshBot

This is Some Information

Information boxes will be used for special information.

This is a Question Box

Questions boxes will be used to ask the reader a question or to think about a concept.

This is an Exercise Box

Exercise boxes will be used to call out something for the reader to do.

Feedback

New revisions of this book are published as significant changes to the PoshBot[2] project are released. If you think a section needs improvement or find something is missing, please post an issue on the ChatOps the Easy Way[3] GitHub repository or contact me via Twitter @devblackops[4].

Current Published Book Version

- Book version 0.1.0
- PoshBot version as of last publication: 0.10.3
- Date as of last publication: 2018-06-16

[2] https://github.com/poshbotio/PoshBot
[3] https://github.com/devblackops/chatops-the-easy-way-examples
[4] https://twitter.com/devblackops

About the Author

Brandon Olin is a Cloud Architect, veteran Systems Engineer, speaker, blogger, freelance writer, and open source contributor. He has a penchant for PowerShell and DevOps processes. He spends much of his time exploring new technologies to drive the business forward and loves to apply ideas pioneered in the DevOps community to the traditional enterprise environment with the goal of solving real-world business problems. Brandon is active in the PowerShell community and loves to give back with a number of projects published to the PowerShell Gallery[5]. He also is a contributor to the IT learning website TechSnips.io[6].

You can follow his code at GitHub[7], his blog at devblackops.io[8], or reach him on Twitter at @devblackops[9].

[5] https://www.powershellgallery.com/profiles/devblackops/
[6] https://www.techsnips.io/
[7] https://github.com/devblackops
[8] https://devblackops.io
[9] https://twitter.com/devblackops

Introduction

Companies of all shapes and sizes are faced with the same problem - innovate or become extinct. Pressure is coming from all directions, challenging organizations to become nimble in an ever-changing digital landscape. Most companies are not in a position to rest on their laurels. They can not assume existing business models will live forever. Disruption is everywhere. They must continuously innovate to stay relevant in their market and to their customers.

Successful companies lean into these challenges and take advantage of new tools and ways of working that put them at an advantage against their competition.

Good is not good enough.

They continually improve. Existing processes are continuously tested, validated, and measured. Small changes are introduced and the effects analyzed. Positives changes are reinforced. Negative changes are abandoned.

This evolution is a never-ending process.

IT is not a just a cost-center to companies facing this challenge head-on. They are not seen as faceless people sitting behind ticket queues where requests go to die. They do not dread having to talk to IT to ask for permission to start a new initiative. They see IT as a partner and enabler for the business.

In these organizations, IT sees their role differently as well. Their role is not to maintain the status quo. Not to maintain the existing systems, keep Service Level Agreements (SLAs) in check, or keep the lights on.

Their role is to help drive the business forward and be willing partners in whatever the business is trying to accomplish. They view their mission as providing **value** to the business. Modern IT organizations are force multipliers. Amplifying business capabilities is what they do.

They are champions of change.

Chapter 1 - Chat: A Brief History

Bulletin Boards

I was on bulletin board systems (BBSs) way back in the day and remember using a 2400 baud modem on my dad's Amiga 1000 to connect to a few local BBSes in the Portland, OR area. I was maybe ten years old and had no clue what I was doing. I would join random chat rooms and be an obnoxious kid trolling and generally annoying people with stupid jokes. Occasionally I would add something constructive to the conversation but in general I just a nuisance. Sometimes I'd find other, like-minded obnoxious kids to converse with, and we'd go off and talk about the usual stuff tweens chat about with each other.

That fact that I could connect my computer over the phone and talk to anyone I wanted was amazing to me. Sometimes I would connect into the chat room and listen, or instead read what others were saying. All kinds of conversations were happening at once, and I would suck it all in.

The excitement with BBSes didn't last forever and once modems increased in speed to 14.4 kbit/s, and later to 28.8 kbit/s, it was the mid 90's and Internet Service Providers (ISPs) were becoming generally available. Services like Prodigy, Earthlink, and America Online (AOL) were becoming household names.

AOL eventually started carpet bombing the United States with "Try America Online" CDs, and the internet began to gain a foothold. More and more people discovered the internet and BBSes quickly started to fade away after that and I moved on to other services.

Early Bot Examples

- ELIZA (TODO)

Modern Chat Bots for Consumers

Most people have heard of Amazon's Alexa, Apple's Siri, Microsoft's Cortana, and Google's Assistant. These are bots in a different form than what we'll cover in this book, but at the primary level, they work the same. They respond to a trigger phrase of some sort, run a command on the back end infrastructure, and return information to the user.

Example triggers for these products:

```
Alexa, set a timer for 10 minutes.
```

```
Hey Siri, how's the weather tomorrow?
```

```
Hey Cortana, remind me to get milk tomorrow.
```

```
Hey Google, play my morning playlist.
```

While these are great for every day uses at a personal level, they are not designed to automate a business and are inherently one-on-one interactions. The core feature of ChatOps is group collaboration and transparency, and that is not possible with one-on-one conversations.

Modern Chat Products

We've moved on from the days of BBSes, and AOL Instance Messenger. Some are still using IRC, but many businesses are now using a group chat tool of some kind. The most popular products out there today for businesses Slack and Microsoft Teams.

Slack

Slack is a cloud-based collaboration and messaging platform that launched in 2013. Started initially as an internal tool used by the founder's previous company Tiny Speck, Slack quickly gained popularity due to its ease of use and the wide variety of integrations to third-party services.

Slack is a freemium model where anyone can create new teams for free. Anyone can join or be invited to these teams which as led to many communities that were

previously using message boards or social media to move to Slack. The free model allows the last 10,000 messages to be viewed and searched and a limited number of integrations at any one time.

Paid versions offer options such as shared channels between teams, unlimited integrations, and group video calls.

Slack's popularity is visible anecdotally. Just look around at laptops during a tech conference, and you'll find half the people have Slack open.

Slack actually stands for **S**earchable **L**og of **A**ll **C**onversation and **K**nowledge.

MS Teams

Microsoft Teams is a new platform from the software giant that launched in 2017. Teams has similar features to Slack but also has deep ties to other Microsoft Office 365 services. Packaged as part of Office 365, Teams is first and foremost a business collaboration tool.

Many companies currently using Skype for Business are taking a hard look at Teams as they often already have an existing Office 365 subscription, and the fact that Teams is packaged as part of their current subscription is attractive.

Given Teams is a relative newcomer, the variety, and breadth of integrations lacks compared to Slack, but this will probably improve over time.

Are Bots Dead?

Chatbots in the consumer space was all the rage for awhile and according to some media reports would take the world by storm. The problem was this idea never really caught on with developers. The media hyped up "conversational commerce" and other terms that positioned bots as a new wave of tools consumers could interact with businesses. This critical mass never happened because developers were never really onboard in the first place.

Conversational UI is hard to get right, and the required Artificial Intelligence (AI) / Machine Learning (ML) capabilities available to most businesses is not there yet. Natural Language Processing (NLP) is an entire area of computer science and without easily consumable frameworks, will be out of reach for most developers. Right now, it is still much more efficient for consumers to interact with a webpage rather than have an open-ended conversation with a bot.

It's important to point out that these types of bots are NOT what this book is addressing. Bots created by companies to support their brand or extend their eCommerce business is not ChatOps.

ChatOps is Different

ChatOps, on the other hand, is not meant to be a consumer-facing technology. ChatOps is all about getting work done inside your business and collaborating with your peers. ChatOps is tailored to your specific requirements and is a way to bring tools inline with conversations for shared context and increased visibility.

In the next chapter, I'll talk a bit more about what exactly ChatOps is and how it can help your organization.

Chapter 2 - Why ChatOps

Definition

ChatOps is a collaboration model that connects people, tools, process, and automation into a transparent workflow. This flow connects the work needed, the work happening, and the work done in a persistent location staffed by the people, bots, and related tools. The transparency tightens the feedback loop, improves information sharing, and enhances team collaboration.

Conversations, put to work

ChatOps puts tools inline with the conversation. It is a way of collaborating that came out of the DevOps movement that unifies communication with actions taken with a goal of removing information silos and facilitating efficient information exchange. This shared context supports a culture of information sharing, transparency, automation, and empathy with co-workers.

These same attributes are related to the DevOps movement as a whole. DevOps is a way of thinking and a way of working with the goal of improving your business, whatever that might be. It focuses on delivering business value faster, safer, and more efficiently. It is about improving human relations, being empathetic to your coworkers, and enabling people to be more effective in their work.

ChatOps embodies this philosophy.

Benefits

The benefits of ChatOps can be distilled down into two categories, social and technical. Different members of the organization may value these differently, but

all have merit and bring benefits in different ways.

Engineers, operators, and other technical staff will naturally gravitate to the technical benefits, but the social impacts of ChatOps cannot be overstated. Remember, DevOps is as much if not more focused on social and cultural change within an organization rather than the pure technical benefits. Management may value the social benefits and how they help facilitate change amongst the company more than the hard technical wins.

The benefits below are just a small collection of the positive behaviors one gains when adapting a ChatOps model.

Social

Cultural Change

It has been said that culture eats process for breakfast. This statement is true and is something you must take into account. The inertia and mass behind a companies' culture will hinder any effort put into changing it. Initiatives to **directly** change culture are fruitless. You cannot change the culture with a simple declaration from senior management. It is more complicated than that. A companies' culture is built up over time and is a product of people behave and empathize with each other. These behaviors have many sources of influence, with tools being just one of them.

Tools can influence behavior because they are one of the primary mechanisms that people use to communicate with each other. Tools that promote open communication across silos will encourage people to do the same.

> You can't change your culture with tools directly, but tools enforce your behavior, and your behavior over time becomes your culture.

Increased Collaboration

ChatOps promotes collaboration by moving communication out of emails and private messages and bringing it into a group chat platform where the conversations

are visible to all. Everyone can benefit and learn from these conversations when walled of information channels are removed.

Knowledge Sharing

Once information is available to all, everyone can learn from it. As people use ChatOps, the conversations and resultant actions become permanent history inside the chat platform. Others can search and look back at this history to discover how a particular problem was solved. Conversational history forms a sort of living documentation that is a collective history of problem-solving knowledge that people can go back and acquire expertise.

Situational Awareness

With conversations and actions in a shared place, everyone has a single window to see what is going on. Problems (and fixes) are out in the open for everyone to see.

You probably have experienced the scenario where a critical incident is occurring, and emails are firing back of forth from people. You may get included on some of these emails and not others. Knowledge shared inside these emails is lost to you, and it may have included valuable information that you could have used to solve the problem.

Email chains are an inefficient way to spread information because often not all relevant parties are included in the conversation. With ChatOps, this communication is consolidated into a single place where anyone interested can view and participate.

Shortened Feedback Loops

Because of this knowledge sharing and situation awareness, feedback loops are shortened. All people relevant to the conversation can pitch in and help. The time it takes to solve a problem is reduced when the information is available to all and people with the knowledge can contribute to the conversation.

Faster Onboarding

Every command issued to the bot is an example for someone else to apply. These serve as breadcrumbs and are useful to new people joining a team because they can

observe how the organization operates by reading back through the chat history. This "day in the life" eases their transition into the team because they can observe how the organization communicates with each other. New hires or others joining the team have living documentation for how common problems are solved within the team, and they can learn how these problems are solved because all the relevant information is available to them inside the chat history.

Team Empowerment

ChatOps democratizes work by making automation available to more people. By allowing others to run commands that traditionally have been solely the role of an engineering team, that helps the other teams to "own their stack." They may not have the in-depth technical knowledge on how to accomplish something regarding their application, but they don't need to. Chat commands are created that allow them to become more self-sufficient in managing or troubleshooting their application without creating tickets for already overloaded engineers to handle.

This "shifting left" way of thinking is about giving the people who possess the most context about an issue the power to solve that issue themselves.

Reduced Lead Time and Cycle Time

Cycle time is the amount of time it takes to complete a task, whereas **lead time** is the amount of time from the moment a person requests work to be done to when the person doing the work notifies them that it is complete.

As a simple example, imagine a customer submits a ticket to create a new virtual machine. This ticket was requested at 9:00 am and sent to the service desk. The service desk reassigns the ticket at 9:30 am to your queue. You do not start working on this ticket right away but instead, decide to tackle it after lunch. At 2:00 pm, you start work on the ticket, and it takes you 2 hours to build the virtual machine, patch it, configure it, and notify the customer that it is complete. By now it is 4:00 pm and the customer has already gone home for the day.

The **cycle time** for this task was 2 hours because that's how long it took you to perform the work but the **lead time** was 7 hours or perhaps even a full day if the requester doesn't see your notice until they come in the next day.

By using ChatOps, both lead time and cycle time is reduced because the requestor can run a command themselves to build a virtual machine. What to them took 7 or more hours can now be reduced to 2 or even less because all unnecessary handoffs are eliminated.

Promotes a Learning Organization

This shared context and living history enables a focus on learning, improving, and evolving services to make them more reliable for the end user. With more people able to participate in managing and troubleshooting services, there are more opportunities for innovation from a broader array of people with diverse technical backgrounds.

Technical

Increased Productivity Through Automation

Automation drives ChatOps. Chat commands execute scripts or binaries that carry out the automation task. This reduction in manual work increases repeatability as automation will perform the same task the same way every time. Processes will be executed quicker and with more repeatability by automation.

Reduced MTTR

With minimal effort, executable runbooks can be created to manage everyday administrate tasks. These runbooks combined with shared context enable more eyes on the problem at hand. When delegating execution of these commands to on-call staff, problems can be remediated quicker without the need for escalation. Having more people able to pitch in and solve a problem reduces the Mean Time To Resolution (MTTR) and provides better service levels.

Better History and Logging

The shared context and permanent history through chat allow a persistent log of actions to be kept, indefinitely if required. Chat history can be used as a learning tool to discover how commands are used or to look back on incidents to determine a timeline of events.

Funneling execution of commands to a shared bot also provides a single mechanism for rich, contextual logging to take place. Security organizations may require the maintaining of accurate and permanent logs of all actions taken, by whom, and the results. With ChatOps and modern group chat platforms, this is easily accomplished.

Improved Safety

ChatOps allows the creating of commands tailored for specific use-cases. It also provides a platform for custom validation logic, authorization, or authentication to be evaluated for particularly sensitive or powerful commands. This customization provides an additional layer of safety, as commands can be written not to perform the action if standards or policies are violated. If the user provides parameter values that you deem unsafe or against policy, you can disallow the command from being executed.

It is recommended to establish these guardrails on any command that has the potential to be misused, both intentionally and unintentionally.

Better Security

ChatOps is in effect, a proxy for users to execute commands. These commands usually are not executed under the users' credentials but instead with system accounts that have the appropriate permissions applied. It is advisable to keep to the policy of least privilege and only grant these system accounts enough access to perform the task they are designed to do and nothing more.

Because commands are not executed under the user's context, command authorization is defined at the chatbot layer. Most ChatOps frameworks have a notion of Role Based Access Control (RBAC) to enable you to control what users can execute what commands.

Common UI/UX Across Tools

ChatOps is an abstraction over commands that interact with infrastructure and services. Each infrastructure component or service will probably have a different mechanism to control it. Instead of the user having to learn and understand dozens or perhaps hundreds of command line tools and administration portals, they can utilize the familiar ChatOps interface. This provides a consistent user interface (UI) and user experience (UX) that users can understand and embrace.

Reduce Handoffs

Because commands can be delegated to anyone using the chat platform, handoffs between teams and silos can be reduced or eliminated entirely. A central infrastructure or tools team may still maintain the responsibility of developing and managing commands, but the execution of those commands are spread amongst the organization. This reduces the burden for development and operations teams who traditionally where the gatekeeper when it comes to administrative tasks. Commands can be delegated to anyone who has a business justification to perform the task. This allows tasks to be accomplished more quickly because there are fewer handoffs between teams. In effect, ChatOps is a better implementation of a self-service portal.

Retrospectives

No matter how redundant or resilient IT systems are designed to be, they are complex, and complex systems always fail eventually. Learning from past outages and discussing how the incident was handled and what improvements can be made is a core DevOps practice. Post-incident reviews, retrospectives, postmortems, etc. are more efficient and provide better outcomes when all the information is available. Having a consistent, shared, and permanent record of conversations and actions taken creates an accurate snapshot of events that can be used post incident for learning and discovery. Think of ChatOps as providing a flight recorder for post-incident review.

Chapter 3 - Platforms, Frameworks, and Use-Cases

While chat has existed for decades, many may be surprised to learn that the concept of ChatOps has been around for years as well. Since the very early days of chat services, the idea of running a command inside the chat window and performing an action has existed. These concepts advanced and evolved over the years, and new tools have entered the ecosystem, but the underlying idea has not changed much.

Common Chat Platforms

Group chat services have existed for many years, but it wasn't until recent advancements in ease of use and capabilities have they become highly popular tools used in most organizations. Most businesses have a group collaboration platform in addition to email and voice services. The following are common chat platforms in use today.

Slack

Slack is a popular group chat platform that has gained wide attention and an ever-growing user base over the recent years. Starting in 2013, it has grown immensely popular due to its user interface and ease of use. Functionally, it is similar to most other group chat platforms on the market.

MS Teams

Microsoft Teams is a recent addition from the software maker. Teams debuted in 2016 as part of the Office 365 product suite and is available to businesses subscribing to the service. Teams integrates heavily with other Office 365 products and is an attractive option for existing O365 users given it is packaged as part of the subscription.

IRC

Internet Relay Chat (IRC) has been around since 1988 and remains a popular choice for many. Its usage has been declining steadily over the years as users have moved to more modern platforms. IRC is not a common group chat platform for businesses as it is primarily used for communication with others on the public internet.

Common Frameworks

There are a wide variety of generic chatbot platforms available today. Most common languages have a bot framework written in them that is tailored for development in that language. Each has similar capabilities in that they all receive a message from a chat network, interpret it, match it to a register command, execute that command, and return the response to the chat network, but each bot handles packaging of commands differently due to language patterns.

Hubot

Hubot is the original chatbot created at GitHub and is where the term ChatOps first originated. Hubot was an internal application written to automate operational tasks for the highly distributed team. Since GitHub was a distributed team, group chat software was a core component to team collaboration there. Overtime Hubot grew to become a critical part of their workflow. Later, Hubot was rewritten and released to the public as an open-source product. Hubot is a Node.js application and is extensible with CoffeeScript or JavaScript. Because it has been around the longest, the number of available scripts and plugins is larger than the other chatbots.

ErrBot

Errbot is a chatbot written in Python and work similarly to the other chatbots discussed. For users proficient in Python, Errbot is an excellent choice when picking a ChatOps tool.

Lita

Lita is a chatbot written in Ruby and uses modules that offer much the same functionality of Hubot. For Ruby developers with a significant amount of automation already written, Lita is probably the easiest to implement.

Cog

Cog is a relative newcomer in the ChatOps space. Cog is engineered to provide many advanced capabilities and is more of a framework than a product. Cog has a significant focus on security and includes a robust access control mechanism. Because of its complexity, Cog has a significant learning curve.

PoshBot

The author of this book created PoshBot in early 2017. It is written in PowerShell and executes the public functions/cmdlets from PowerShell modules. Ease of use and low friction command creation in the primary motivator with PoshBot. The rest of this book will focus on the features and use-cases for PoshBot.

Third-Party Integrations

Group chat products usually provide an application directory of some kind where third-party integrations can be connected or configured. These integrations provide add-on functionality that is specific to the integration author. The use of integrations reduces the need for custom chatbots or commands as the services interface directly with the chat application.

Because there is a wealth of existing integrations available for all the major chat applications, new functionality is easily added without custom development.

Third-party integrations can be limited in what capabilities they provide or the data they return. Because of this, many people turn to chatbots where they have ultimate control over how the command operates and what data is returned.

The benefits of integrations should not be overlooked though. They are an easy and low friction way to get started with ChatOps and help build momentum in adopting the patterns.

Common Use-cases

ChatOps provides a standard interface to the various tools in use in an organization. Interactions with these tools along with the shared conversations provided by group chat applications are one of the primary benefits of ChatOps. Most existing tools have an API that can be utilized with minimal development effort and exposed through the chatbot to provide that common UI layer for teams to consume.

So what are some common use-cases for ChatOps? In this section, we'll discuss several examples of where ChatOps is beneficial.

Notification Funnel

Using group chat as the mechanism for shared context with conversations is a core feature of ChatOps. Conversations and actions taken inline with each other provide the full picture of events as they happen. Pushing events or alerts from existing applications into group chat is another way of centralizing this context for visibility to more parties. Most chat applications provide an API for submitting messages programmatically. Integrating these services with group chat provides even more context as alerts or notification events are often the initiators for additional conversations and actions to be taken by users.

Examples of Notification Funnels

- Infrastructure alerts
- Change or deployment events
- New service desk tickets
- Version control repository commits

Retrieval of Data

Retrieving read-only data about the state of the environment is a common first step when organizations being to implement ChatOps. Providing repeatable and safe

commands to retrieve information increases visibility into the state of systems and enables more in-depth conversations to be had or further actions taken. Retrieval of data without directly giving people access to backend databases or systems empowers the team to share information without consequence.

Examples of Retrieving Data

- Querying infrastructure state
- Getting service desk ticket status
- Query Active Directory information
- Pull metrics or logs from monitoring tools
- Retrieve reports from analytics systems

Modifying Infrastructure

Once ChatOps has been used to centralize notifications and provide useful read-only data for team context, commands that modify infrastructure state or perform other administrative actions is where the real power of ChatOps is. Most development and operations teams have existing automation scripts used for daily administrative tasks. Exposing this automation through the chatbot elevates ChatOps from merely a helper platform to a core enabling technology.

Chatbots decouple the user from the automation that is executed. Because of this, users can be granted authority to run automation to which they would not typically have access. Commands can be created to execute with appropriate guardrails put in place to allow them to be performed by staff who do not (and should not) have the administrative access.

This delegation empowers users to get their work done faster and more efficiently and also frees up technical teams to focus on more important tasks rather than fielding time-consuming work requests.

Examples of Modifying Infrastructure

- Provisioning virtual machines
- Restarting application services
- Executing application runbooks
- Active Directory management
- Database management
- Updating application configuration

The Sky is the Limit

ChatOps is all about enabling humans to work faster, more efficiently, and safer. When implementing ChatOps, examine existing work processes that are time-consuming, inefficient, and can be quickly provided to other teams to execute. There is virtually no limit to what tasks can be created through ChatOps. These empower users to get work done faster and in a place that provides shared visibility and context to the team.