

Lecture Notes for Lecture 11 of CS 5500
(Foundations of Software Engineering) for the
Spring 2021 session at the Northeastern
University San Francisco Bay Area Campuses.

Managing and Tracking Communications

Philip Gust,
Clinical Instructor
Department of Computer Science

*Information and examples in this lecture are based on
recommended readings.*

<http://www.ccis.northeastern.edu/home/pgust/classes/cs5500/2021/Spring/index.html>

Managing and Tracking Communications

Review of Lecture 10

- In lecture 10, we began discussing managing and tracking the development process, enabling teams to assess progress against goals and estimates, and make adjustments as soon as possible.
- This lecture was about managing and tracking project resources: developer effort and time to complete a project by
 - ***planning*** ways to utilize the resources that minimize the resources required to complete the project.
 - ***tracking*** the development process and determining whether the use of the resources continues to match the plan
 - ***revising*** the plan by adjusting utilization of the resources, the amount of resources available, and/or the requirements, in a way that satisfies the needs of the customer.

Managing and Tracking Communications

Review of Lecture 10

- We examined a common myth that adding more developers to a project that is falling behind can help it catch up. However, that has a disruptive effect that can cause schedules to slip further.
- In fact, we were surprised to learn that reducing the number of developers and adding a little more time instead can be a more effective strategy.
- Using the Putnam-Norden-Rayleigh (PNR) curve, relating effort and delivery time, and Putnam-Meyer's software equation, we saw that adding just 25% more time could halve the developers required.

Managing and Tracking Communications

Review of Lecture 10

- We were introduced to task networks to identify dependencies among tasks, and the concept of critical tasks and the critical path through a dependency network.
- Based on task networks, we learned that PERT and CPM scheduling methods use planning information already available in the form of work breakdown structures (WBS) to represent tasks.
- We also saw a time-line view of the same information, a Gantt chart, that can present as a chronological view of tasks, with some performed concurrently others sequentially.

Managing and Tracking Communications

Review of Lecture 10

- Finally, we learned about two special techniques for managing and tracking resources. Time-boxing can be applied when a project cannot be completed its original delivery date.
- With time-boxing, each task gets a fixed amount of effort before moving to the next one. The idea is that a significant amount of each task can be completed, and the rest can be delivered later.
- Earned value analysis is a quantitative technique that helps track progress. The total effort as a percentage is divided among the tasks to help measure “percentage of completeness.”
- Earned value analysis gives a surprisingly accurate and reliable reading of performance as early as 15 percent into the project.

Managing and Tracking Communications

Introduction

- In this lecture, we will continue to discuss managing and tracking the development process.
- Managing and tracking software development enables teams to assess progress against their own goals and estimates, and to make adjustments as soon as possible to stay on track.
- As we learned earlier, measurement is an important aspect of estimation, and managing and tracking the software development process provides information to improve future estimates.
- In this lecture, we will explore managing and track project communications. In the next lecture, we will look at managing and tracking the development process

Managing and Tracking Communications

Introduction

- As we learned from lecture 9, communication is key in a software project. For successful project execution, effective communication to all stakeholders is essential.
- In this lecture, we will look at methods and representative tools for managing various kinds of communications, and for monitoring communications and measuring its effectiveness.

Managing and Tracking Communications

Introduction

- Communications in a project take place across constituencies who have an interest in the outcome, including
 - **development team** who are responsible for initiating, running, and managing the development process.
 - **stakeholders** who are not directly involved in the day-to-day process, but who have a direct influence on the success of the project.
 - **customers and end-users** who decide to adopt the product being developed and to become users of the product.
- Each have specific communication requirements, and specific ways that they prefer communications to occur.

Managing and Tracking Communications

Development team communications

- Communications requirements within the development team are the most complex for a number reasons:
 - the duration of the project requires many modes of communication because of the many types of information that must be shared.
 - communication patterns are complex because many channels of communications are active at the same time
 - the information being communicated is inherently complex, including technical information, logistical information, and social information
 - the durability of the information also varies widely, from informal and ephemeral, to formal and of lasting importance

Managing and Tracking Communications

Stakeholder communications

- Requirements for communications between the development team and stakeholders can vary in complexity, depending on the nature of the relationship.
 - For stakeholders who have close working relationship with the development team, communications patterns may be nearly as complex because of frequency and complexity of interactions.
 - For stakeholders with a more detached relationship, the frequency and complexity may be significantly less, and communications may be more formal and have fewer modes of interaction.

Managing and Tracking Communications

Customer and end-user communications

- The relationship between the development team and their customers and end-users are often the least complex because the information communicated more oriented toward results than process.
 - Customers tend to be interested in limited information around the logistics of adopting a product, including cost and schedules. Communication can be detailed, but infrequent and more formal.
 - End-users also tend to be most interested in limited kinds of information, but around product features, training, and support. Communication is less detailed, less frequent, and even more formal

Managing and Tracking Communications

Types of communication

- We learned earlier that communications among the development team, and with stakeholders, customers, and end-users can be characterized along two dimensions: mode and frequency.
 - **Written Communication:** It is one of the most precise forms of communication that is transmitted via a correspondence medium. It can be further segregated into two forms:
 - *Written Formal:* Project charter, scope statement, project plan, work breakdown structures, project status, complex issues, contract related communications, memos etc.
 - *Written Informal:* email, notes, letters, regular communication with team members etc.

Managing and Tracking Communications

Types of communication

- We learned earlier that communications among the development team, and with stakeholders, customers, and end-users can be characterized along two dimensions: mode and frequency.
 - **Oral Communication:** This type of communication has a high degree of flexibility is done through the medium of personal contact, the team meets, telephonic etc. It can be further categorized into two forms:
 - *Oral Formal:* Presentations, speeches, negotiations etc.
 - *Oral Informal:* Conversation with team members, project meetings, break-room or war-room conversations etc.

Managing and Tracking Communications

Types of communication

- We learned earlier that communications among the development team, and with stakeholders, customers, and end-users can be characterized along two dimensions: mode and frequency.
 - **Non-Verbal Communication:** This is the most basic form of communication.
 - Approximately 55% of communication is done in this form.
 - General examples of this type of communication are facial expressions, hand movements, posture, tone of voice while speaking, etc.

Managing and Tracking Communications

Types of communication

- Another type of communication is targeted at the development team, though the sender is not human, but an automated system.
- As more automation is used to support the development process, it is useful for those systems to deliver information in the form of communications delivered to channels normally used by humans.
- Examples include:
 - a text message about a change in status of a reported defect
 - a desktop notification that an automated build has completed
 - a voice message reporting on a unit test that has failed
 - email with a daily report on issues being tracked
- These communications would have previously come from IT staff.

Managing and Tracking Communications

Unifying communications

- Messages can be sent using multiple channels of communications. For example, the development team might use a combination of channels, depending on what information is being communicated.
- A text message might be appropriate for short, informal questions. However, an interactive discussion may be better conducted using an interactive chat program or conferencing system.

Managing and Tracking Communications

Unifying communications

- Information communicated through one channel of communication is generally not available through other channels.
- Channels of information that are isolated and difficult to navigate are referred to as *silos*, after the cylindrical structures where farmers store grain after harvest. There is no easy way to find and access communications stored across silos.



Managing and Tracking Communications

Unifying communications

- A class of products known as Unified Communication Systems (UCS) seeks to integrate information exchanged through multiple communication channels.
- A UCS provides services that are useful to software development teams:
 - Captures and tracks communications across channels
 - Provides a common way to search for information across channels
 - Archives communications for long-term storage and retrieval
 - Offers data analytics that enable communications to be analyzed



Managing and Tracking Communications

Unifying communications

- UCS products use one of several integration strategies.
 - Allow information to exist in multiple silos but provide a common index that identifies places where the information is stored, enabling the communicated information to be searched.
 - Gather renditions of information stored across silos into a single silo that not only enables the information to be searched, but also provides a useful rendition of the information.
 - Integrate the channels into a communication framework with a common storage scheme that presents renditions of the information using different “edge devices” such as IM, email, voice, etc.

Managing and Tracking Communications

Unifying communications

- It would be difficult for any UCS product to provide all functionality required by a business or a development team. That is why many provide ways to automate and integrate with external products.
- The primary mechanisms provided by UCS products are:
 - **Hooks** to receive events or communications from external products
 - **Integration points** for sending events or communications to external products
 - **Scripting** or other automation mechanisms for adding new behaviors that communicate with external products

Managing and Tracking Communications

Example: Slack

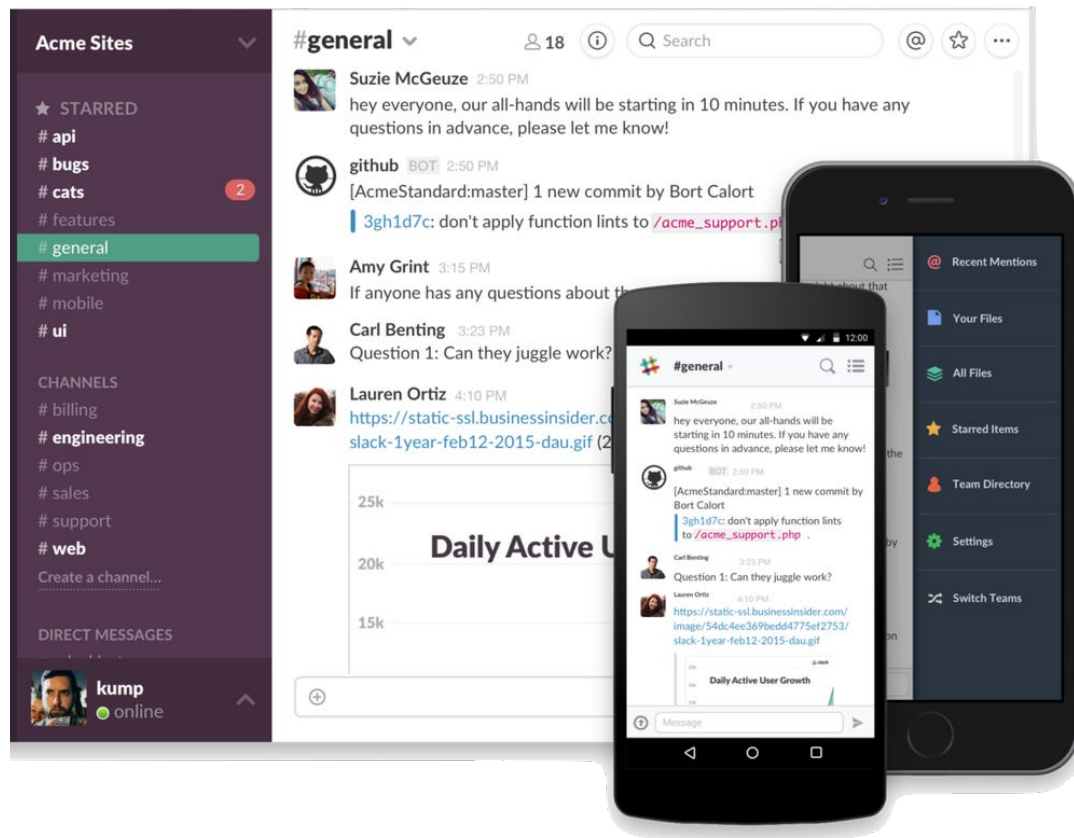
- Slack is a cloud-based UCS that can be accessed through a web interface, or client applications that are available on desktop and mobile devices. A free version offers a limited-functionality for small teams.
- Slack is a product from Slack Technologies, Inc. It began as an internal tool at a company run by one of its creators and became a commercial product in 2013. The company went public in 2019.
- In December 2020, Salesforce.com announced that it reached an agreement to acquire Slack for \$27.7bn in cash and stock.
- The product name is said to be a *backronym* for “**S**earchable **L**og of **A**ll **C**onversations and **K**nowledge.”
- Note: The instructor has no financial interest in this company. It is presented here to demonstrate UCS implementation strategies.



Managing and Tracking Communications

Example: Slack

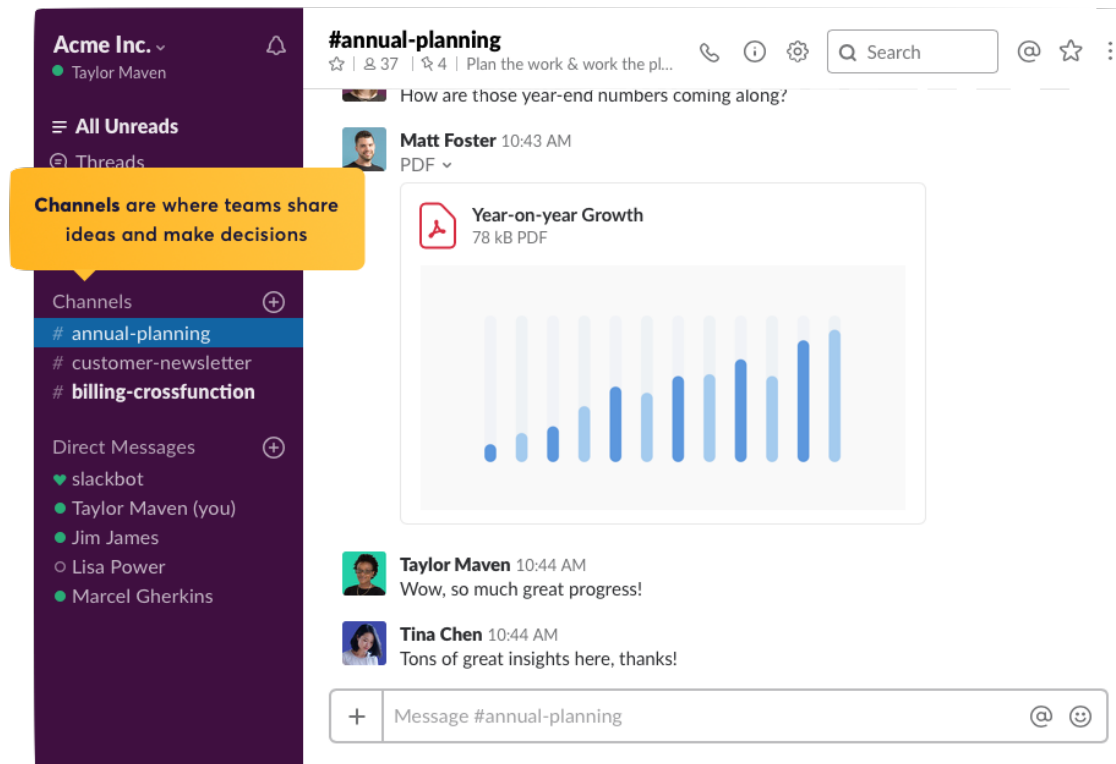
- Slack presents a familiar discussion group metaphor and integrates with many software development tools.



Managing and Tracking Communications

Example: Slack

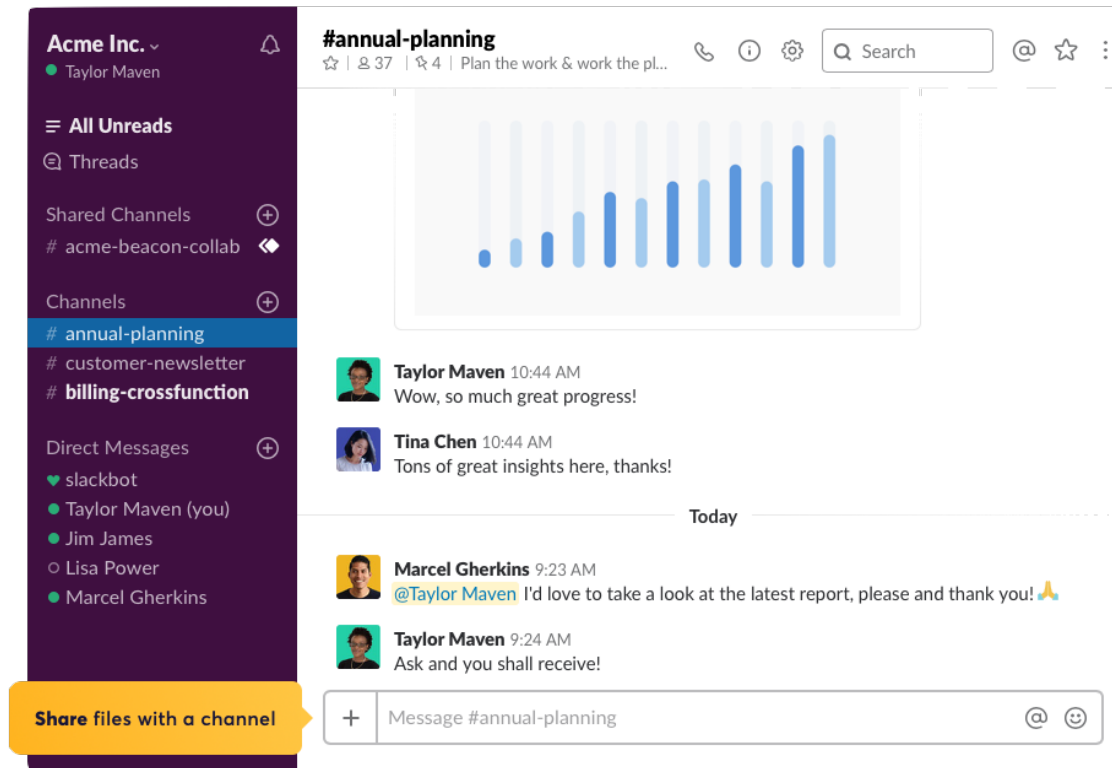
- Slack organizes activities into *channels*. A team can create many channels. For example, a department can have its own channel.



Managing and Tracking Communications

Example: Slack

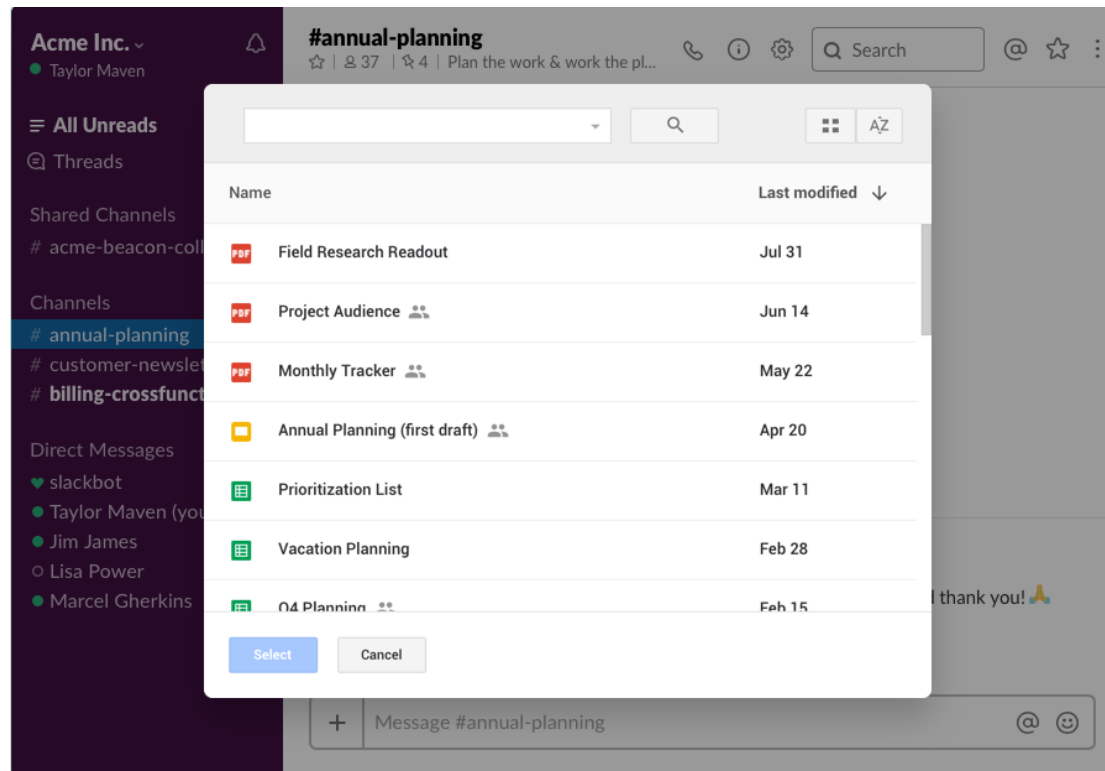
- Slack allows you to share files within a channel.



Managing and Tracking Communications

Example: Slack

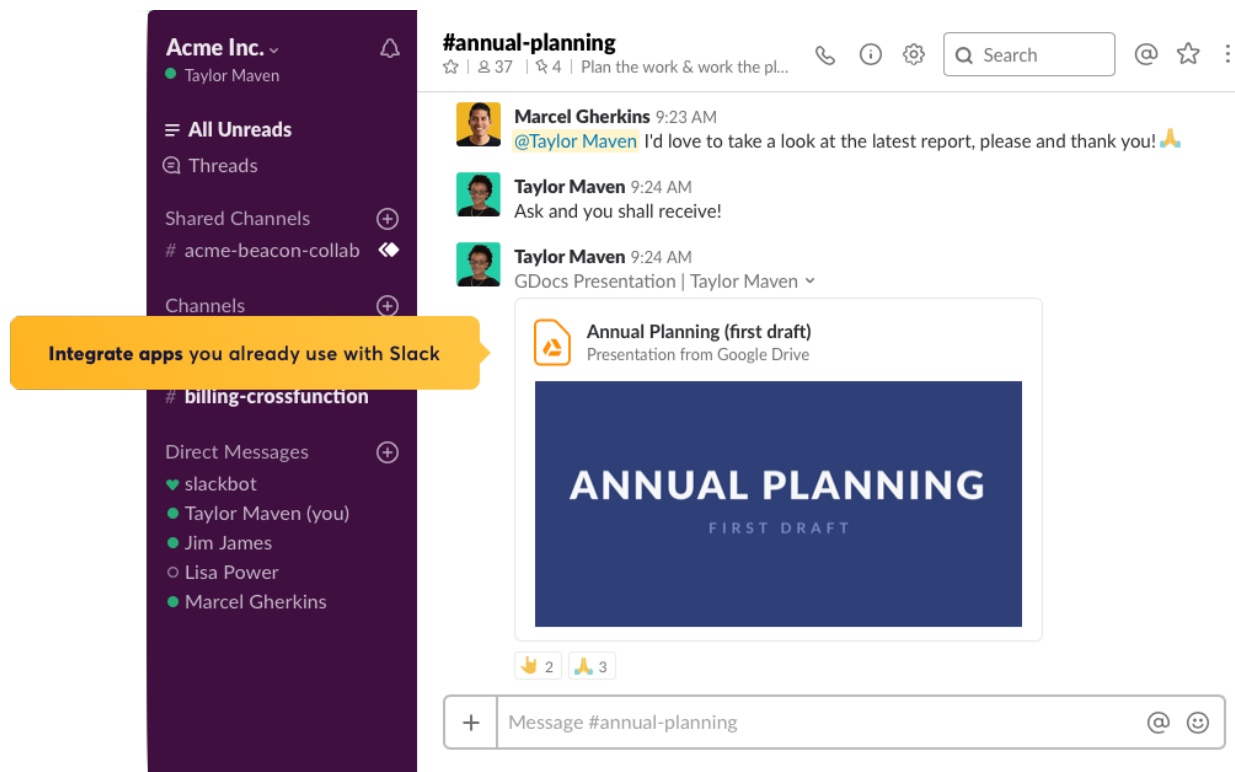
- Slack allows you to share files within a channel.



Managing and Tracking Communications

Example: Slack

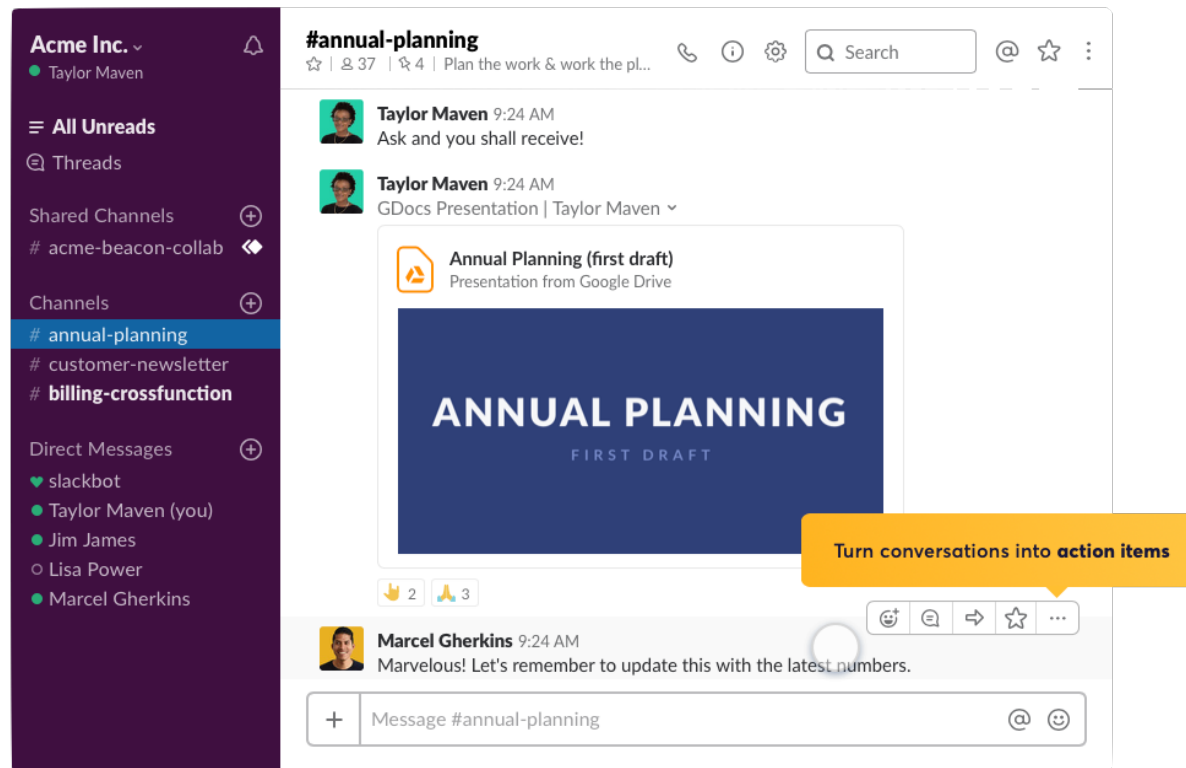
- Shared files can come from internal or external sources.



Managing and Tracking Communications

Example: Slack

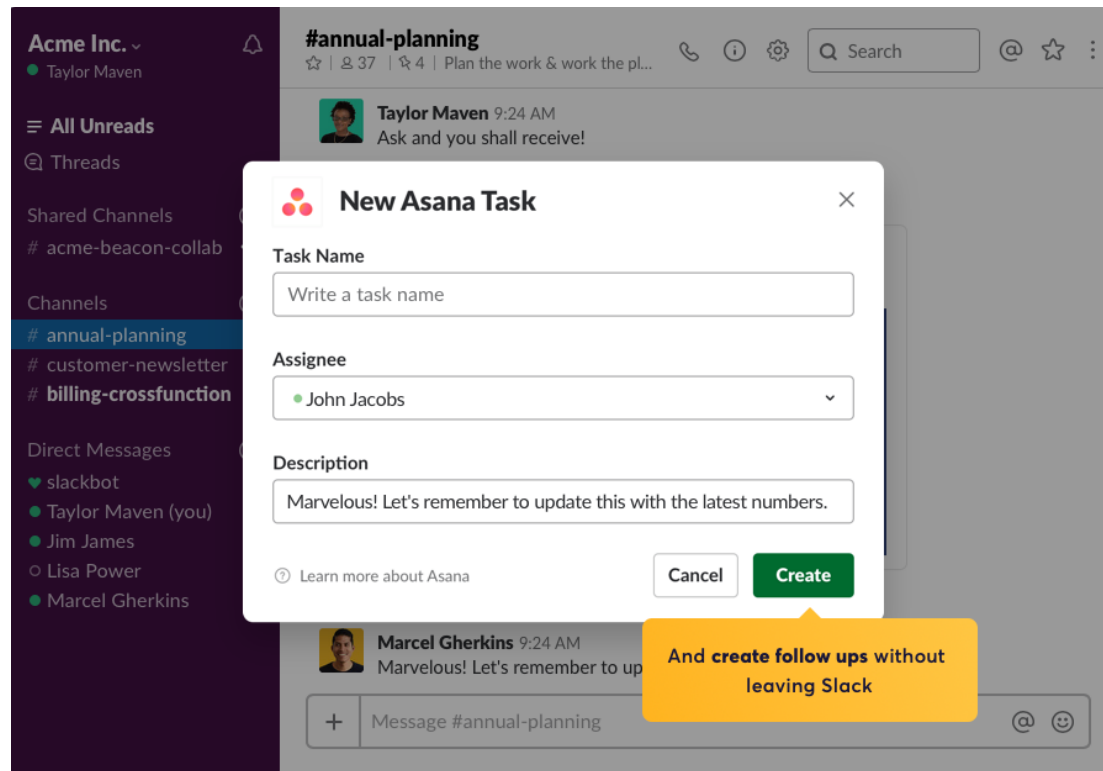
- Slack allows creating action item tasks for a conversation.



Managing and Tracking Communications

Example: Slack

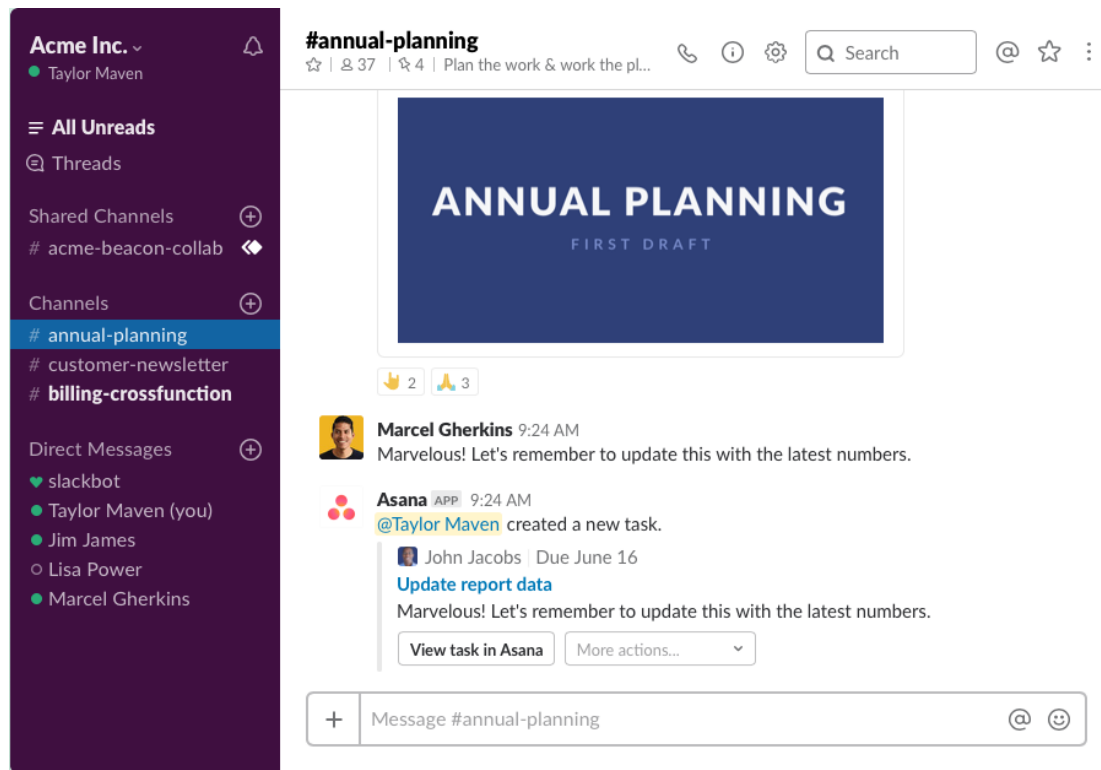
- Slack allows creating action item tasks for a conversation.



Managing and Tracking Communications

Example: Slack

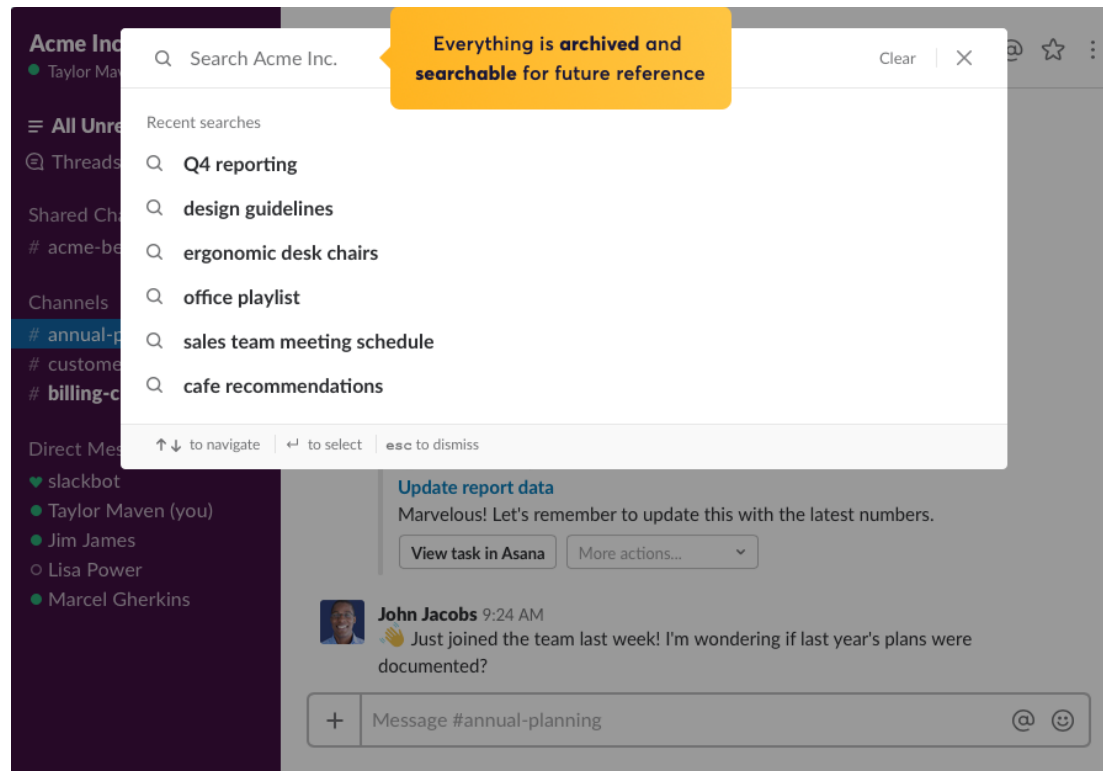
- This new task reminds team members to update the plan.



Managing and Tracking Communications

Example: Slack

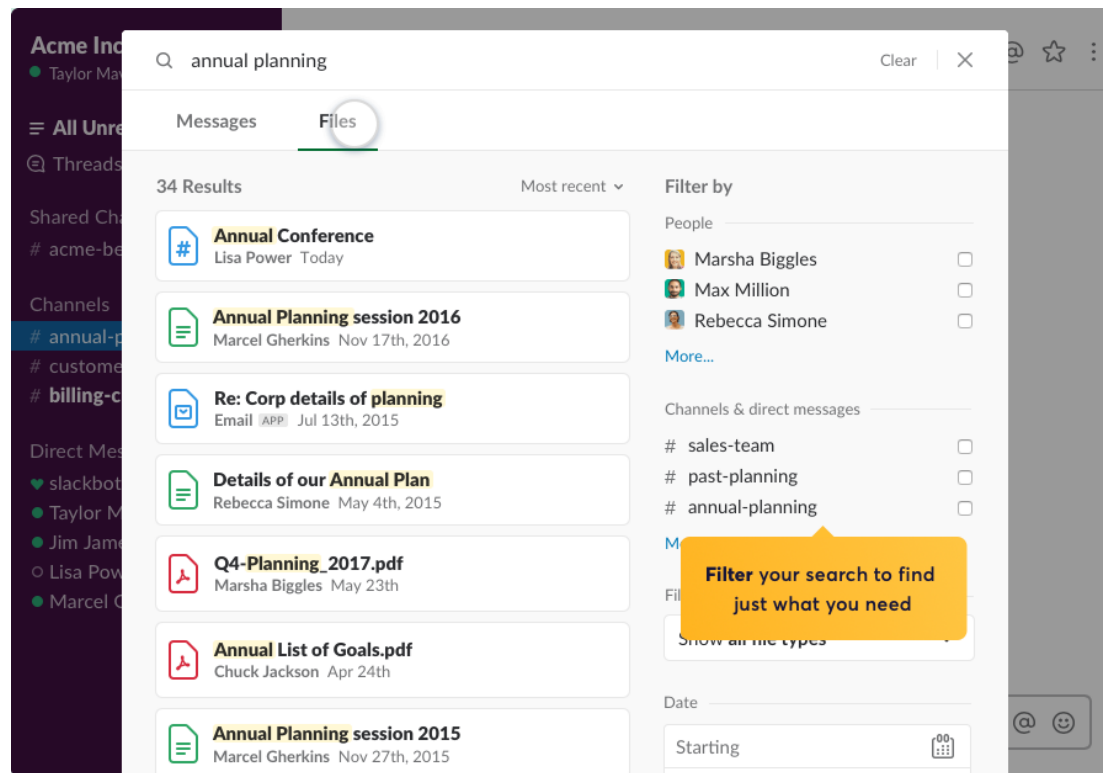
- Slack allows searching for archived content.



Managing and Tracking Communications

Example: Slack

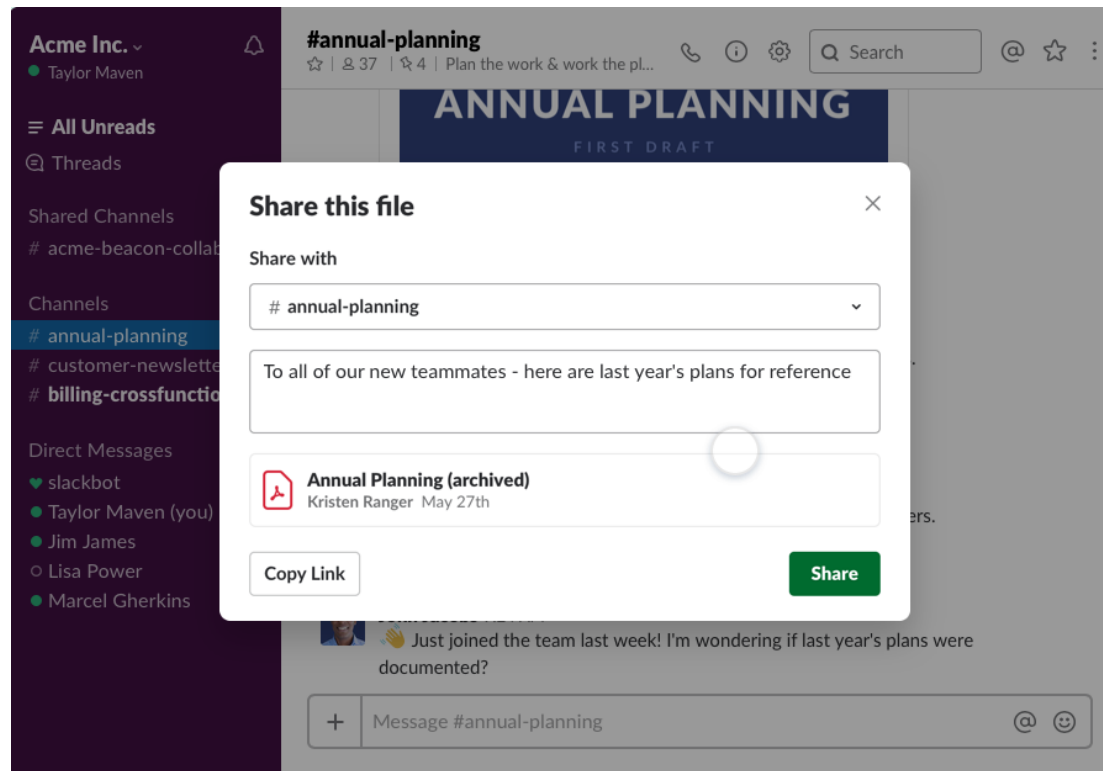
- Slack allows searching for archived content.



Managing and Tracking Communications

Example: Slack

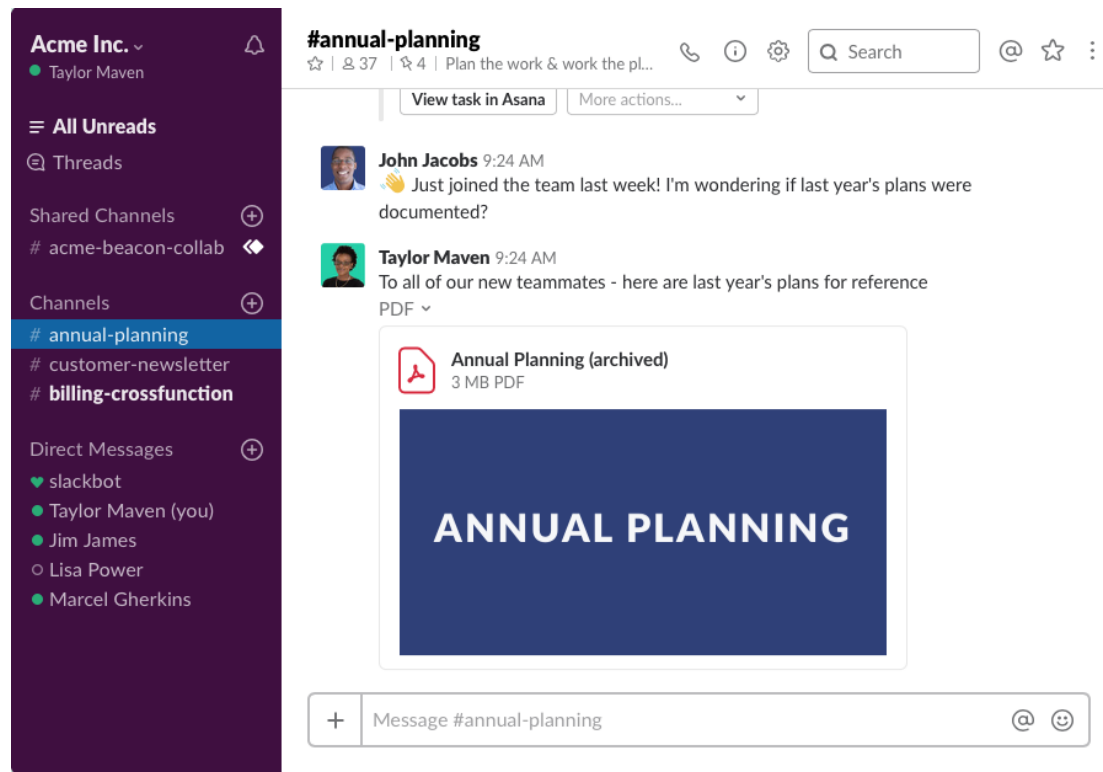
- Archived content can be added to a conversation.



Managing and Tracking Communications

Example: Slack

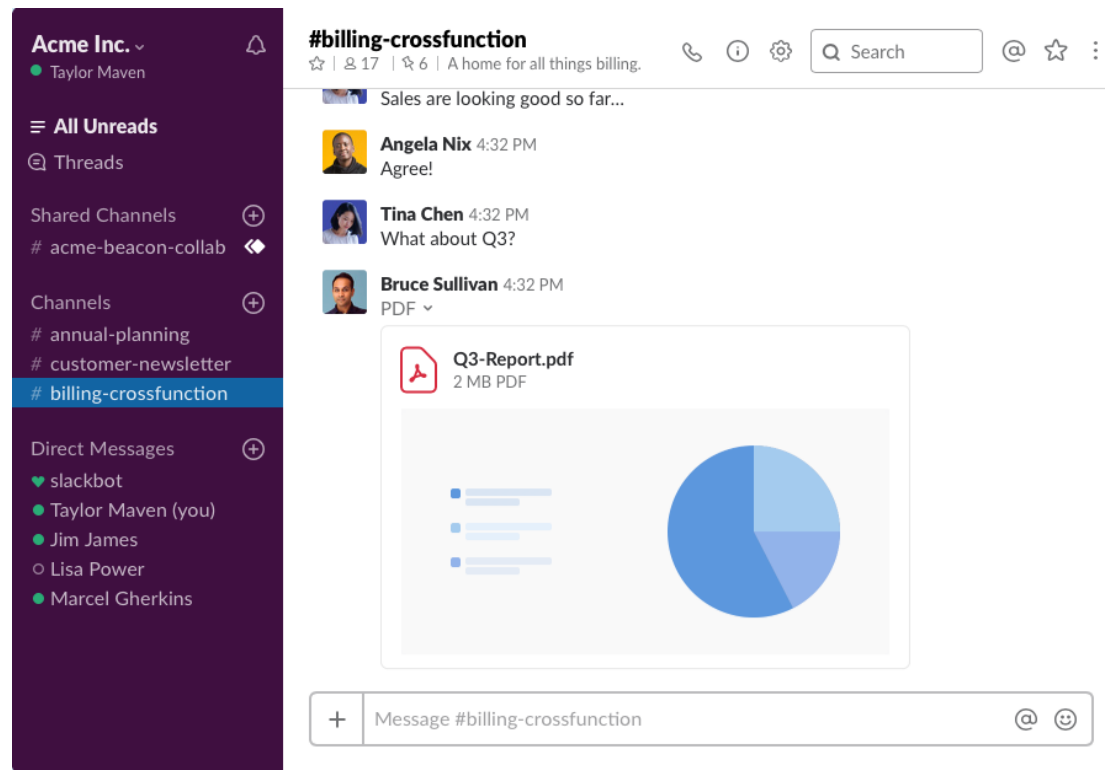
- Archived content can be added to a conversation.



Managing and Tracking Communications

Example: Slack

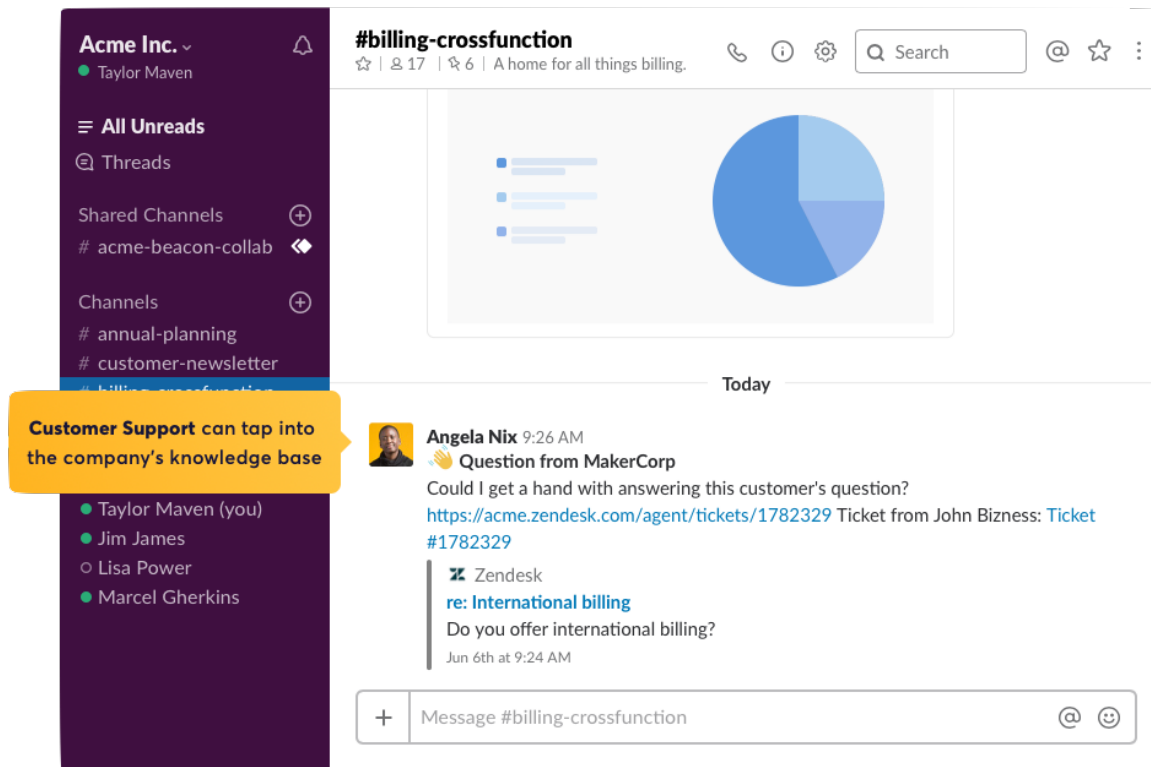
- Slack allows collaboration across functional teams.



Managing and Tracking Communications

Example: Slack

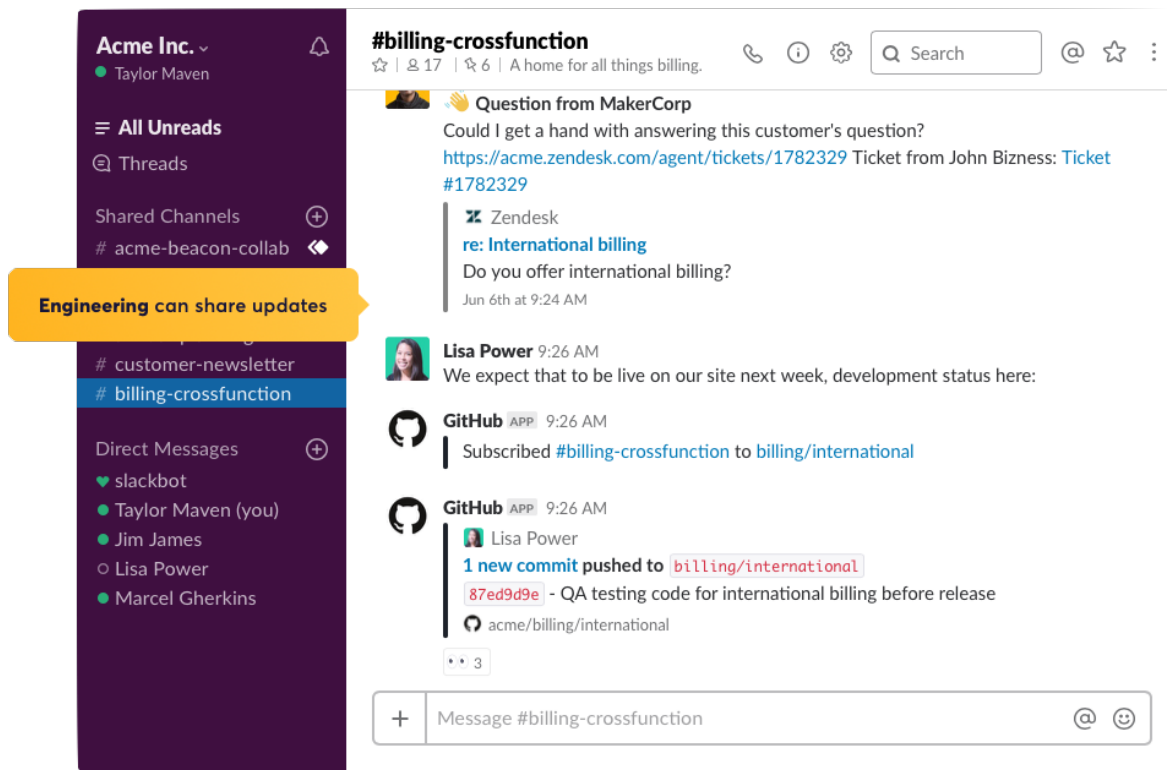
- Slack allows collaboration across functional teams.



Managing and Tracking Communications

Example: Slack

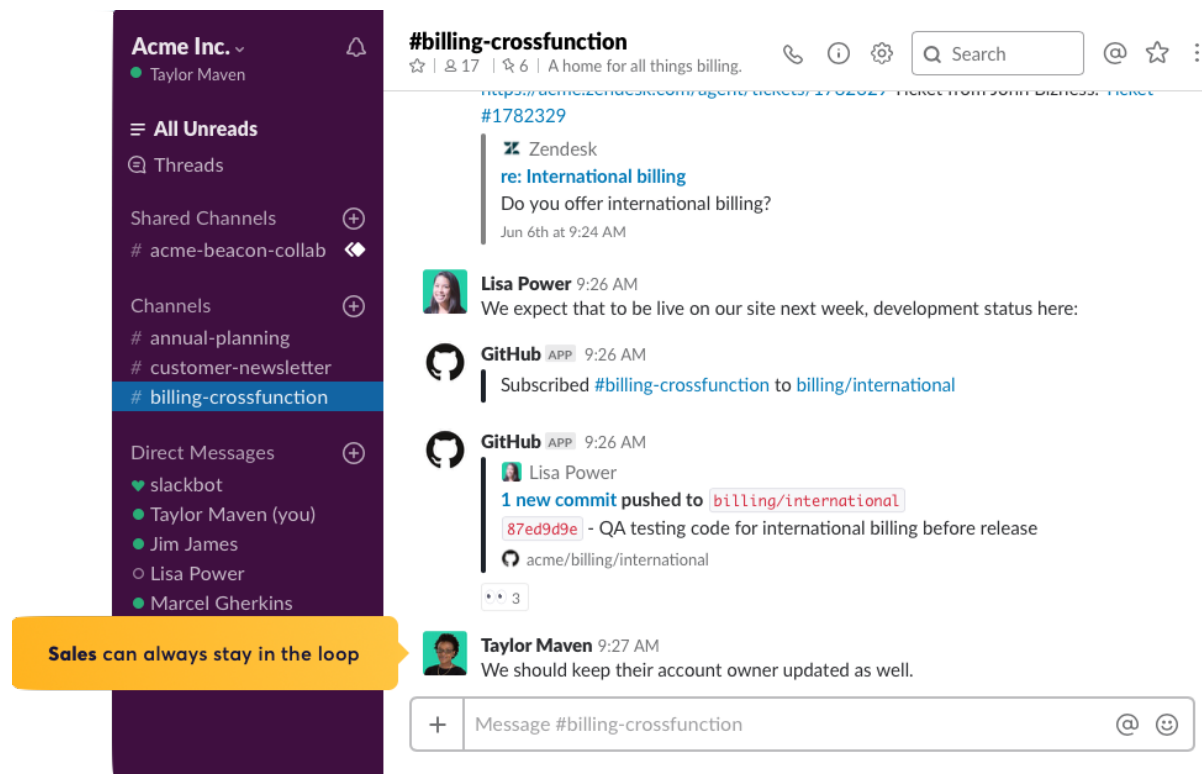
- Slack allows collaboration across functional teams.



Managing and Tracking Communications

Example: Slack

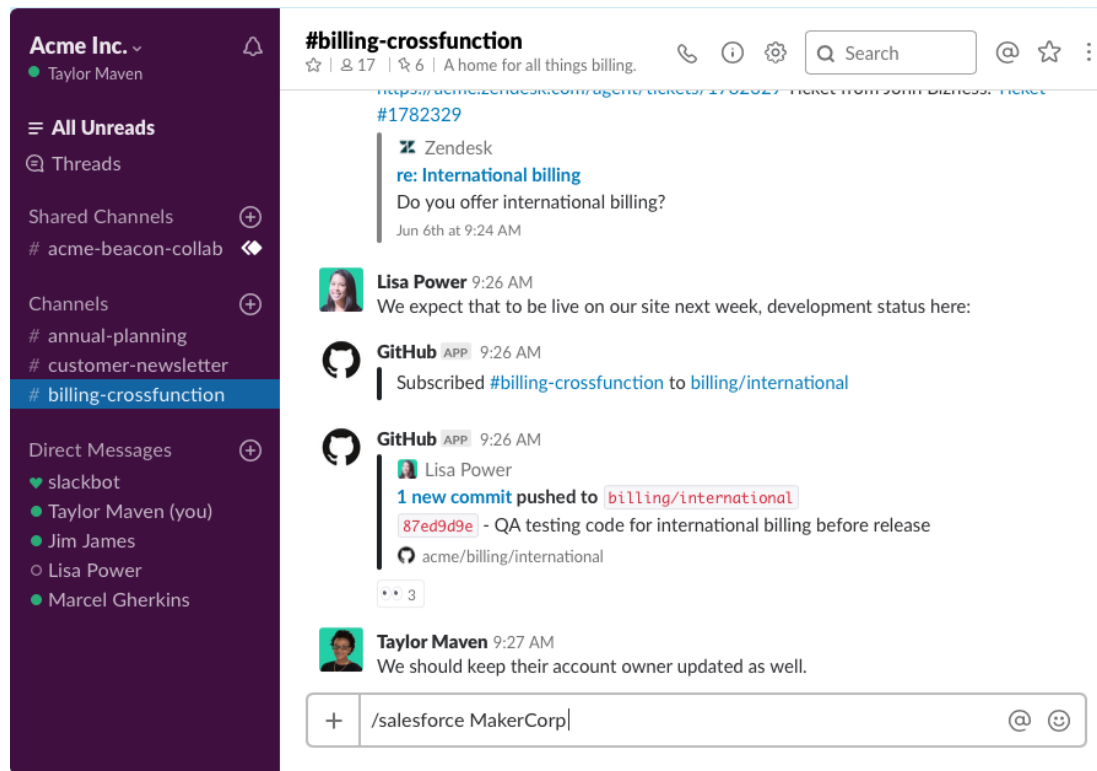
- Slack allows collaboration across functional teams.



Managing and Tracking Communications

Example: Slack

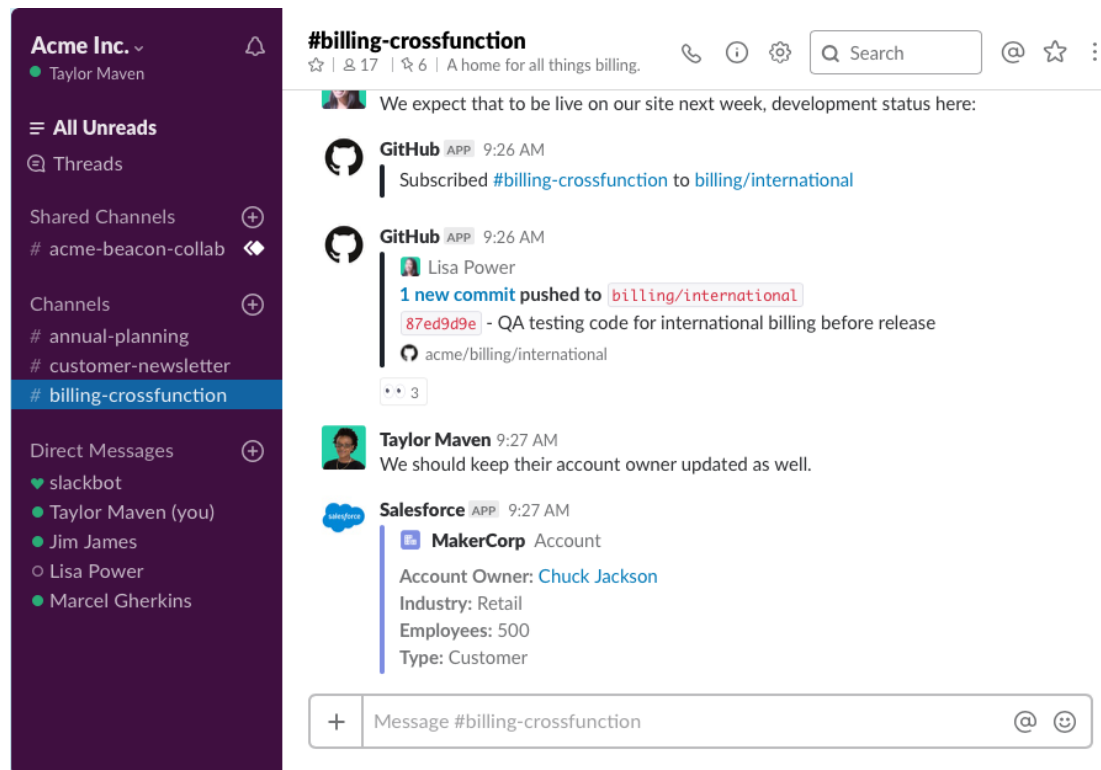
- Slack allows collaboration across functional teams.



Managing and Tracking Communications

Example: Slack

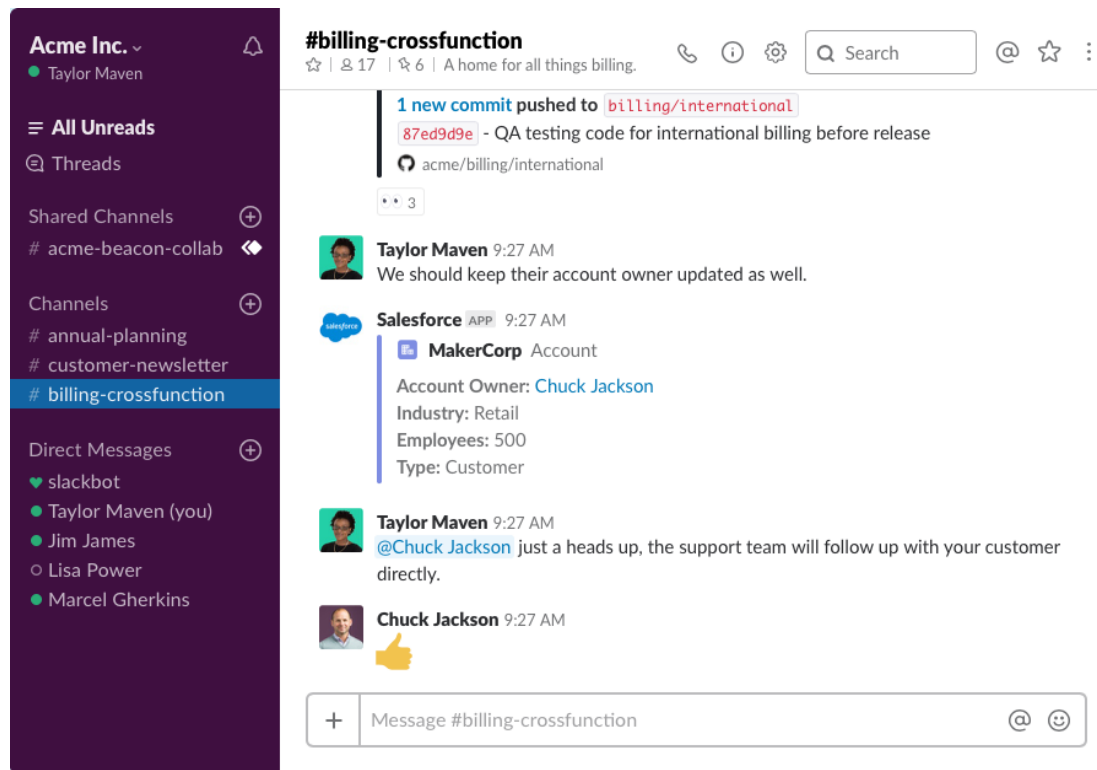
- Slack allows collaboration across functional teams.



Managing and Tracking Communications

Example: Slack

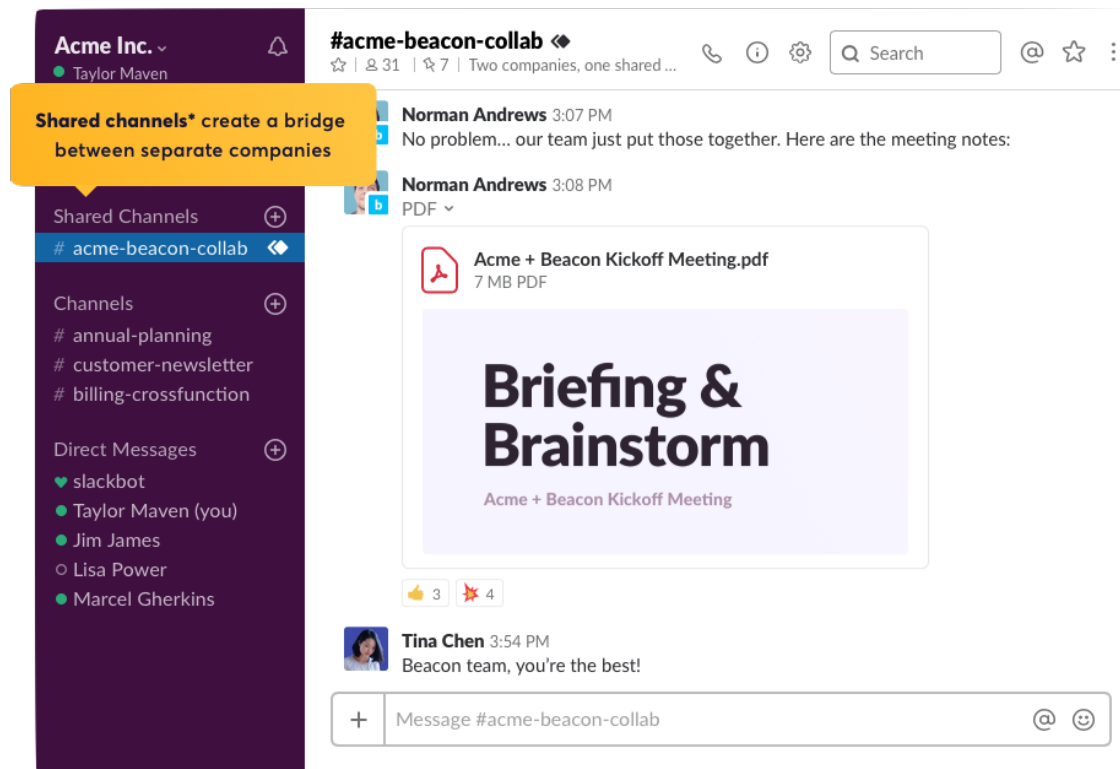
- Slack allows collaboration across functional teams.



Managing and Tracking Communications

Example: Slack

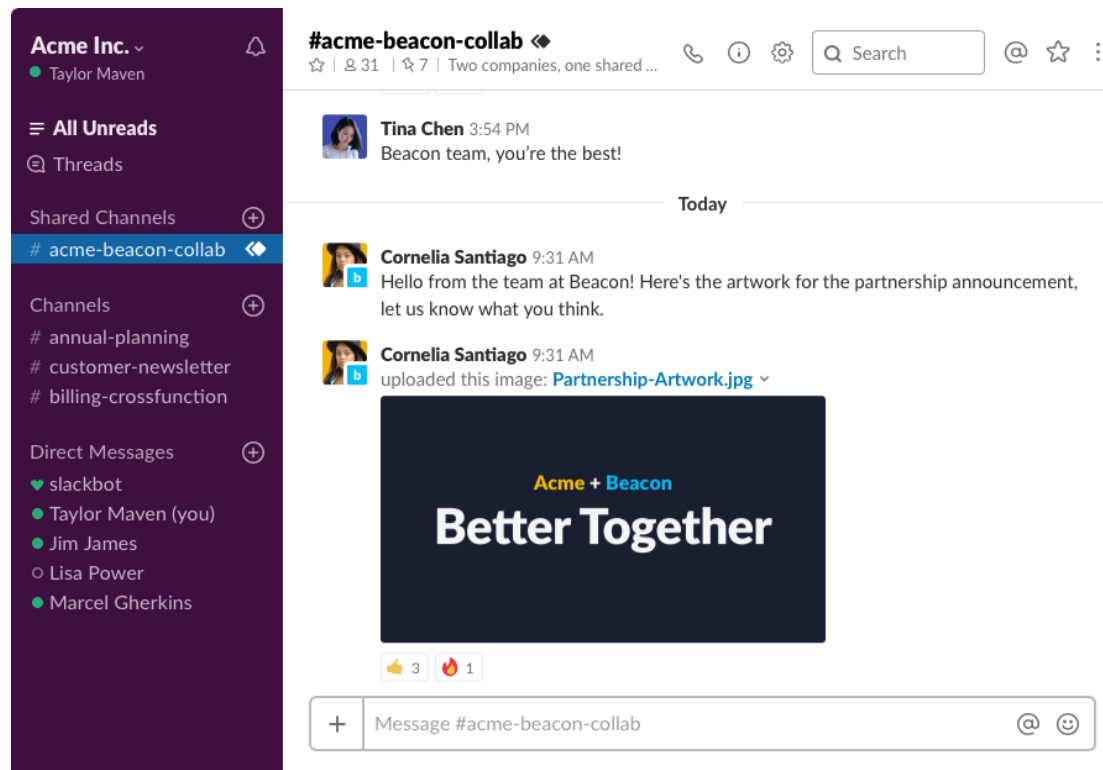
- Slack allows collaboration across companies with shared channels.



Managing and Tracking Communications

Example: Slack

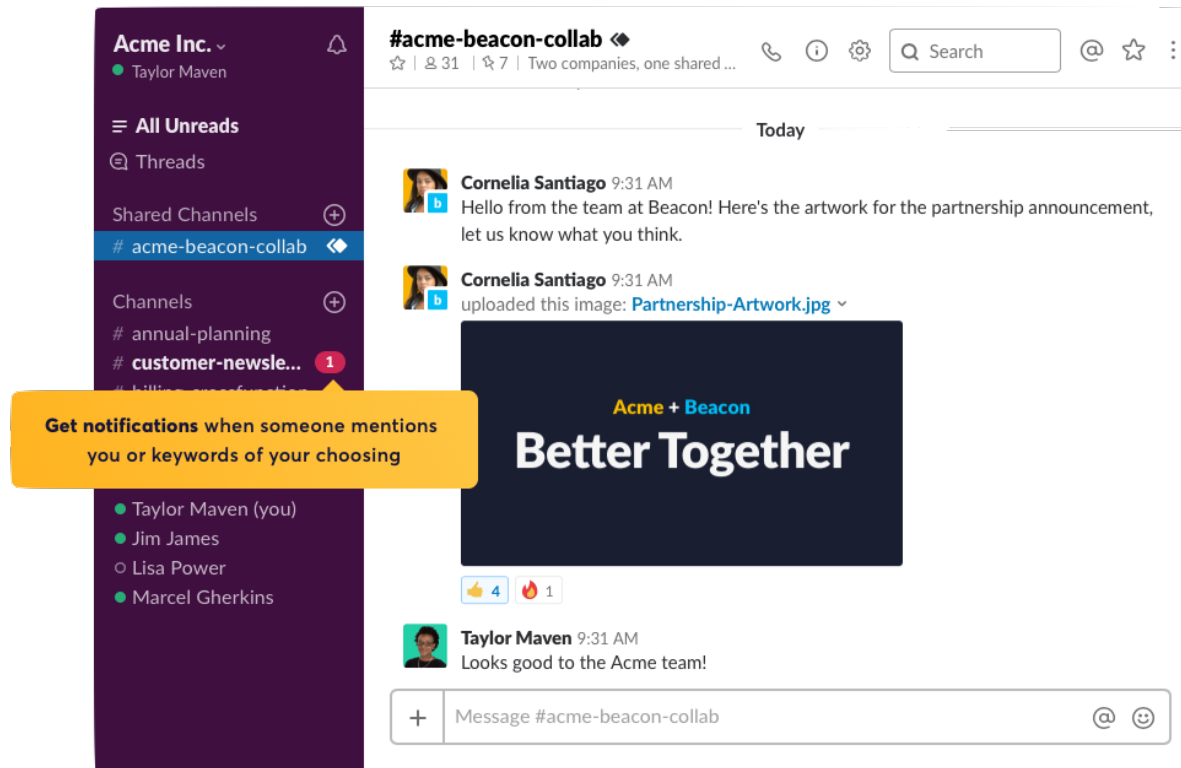
- Slack allows collaboration across companies with shared channels.



Managing and Tracking Communications

Example: Slack

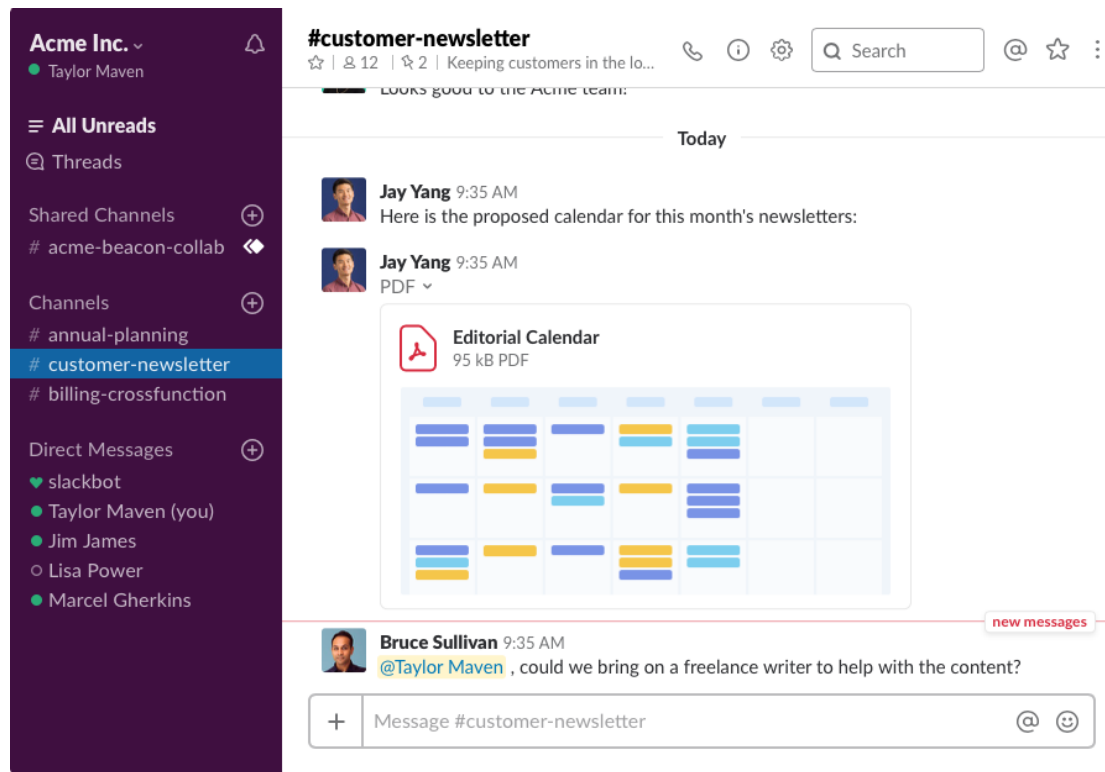
- Slack can notify if someone mentions you or a chosen keyword.



Managing and Tracking Communications

Example: Slack

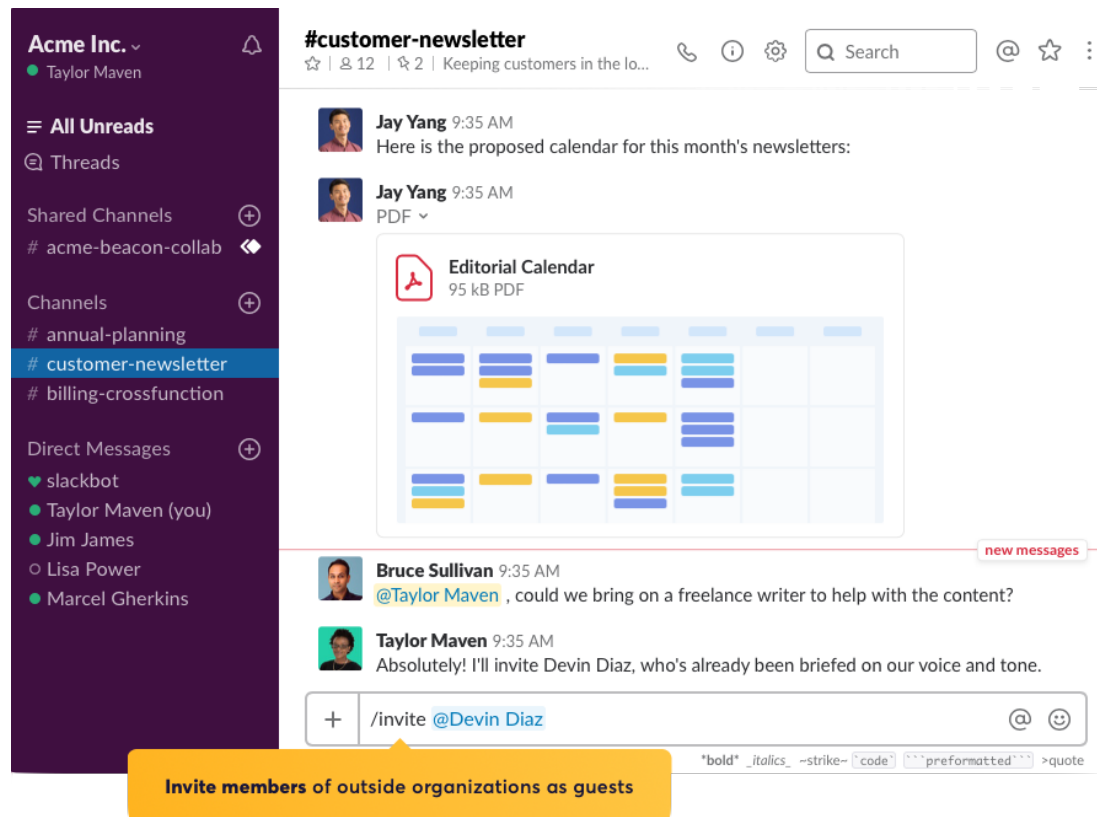
- Slack can notify if someone mentions you or a chosen keyword.



Managing and Tracking Communications

Example: Slack

- Slack supports outside collaborators.

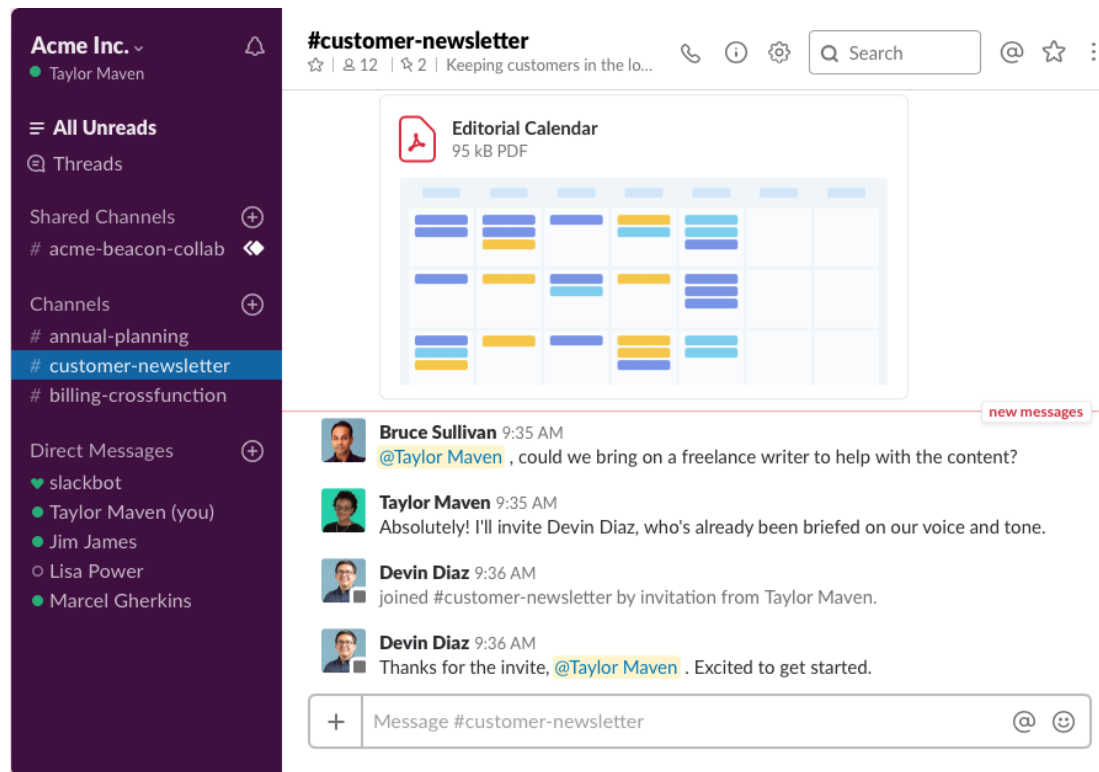


The screenshot displays a Slack interface for a channel named **#customer-newsletter**. The left sidebar shows the workspace **Acme Inc.** with a list of channels including **#customer-newsletter**, which is currently selected. The main channel view shows a message from **Jay Yang** at 9:35 AM, stating "Here is the proposed calendar for this month's newsletters:" and attaching a PDF file titled **Editorial Calendar** (95 kB). Below this, a message from **Bruce Sullivan** at 9:35 AM asks, "@Taylor Maven, could we bring on a freelance writer to help with the content?". Taylor Maven responds at 9:35 AM, "Absolutely! I'll invite Devin Diaz, who's already been briefed on our voice and tone." The bottom of the interface shows a text input field with the command **/invite @Devin Diaz**. A yellow callout box at the bottom center of the screen reads: **Invite members of outside organizations as guests**.

Managing and Tracking Communications

Example: Slack

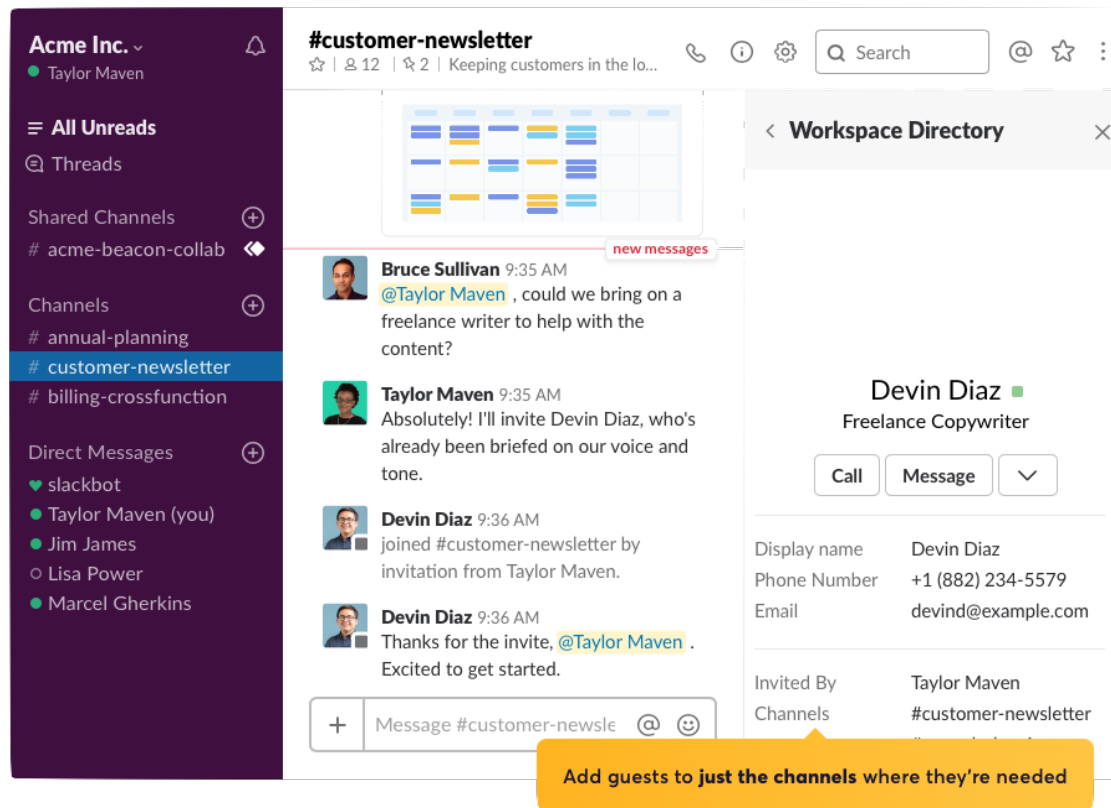
- Slack supports outside collaborators.



Managing and Tracking Communications

Example: Slack

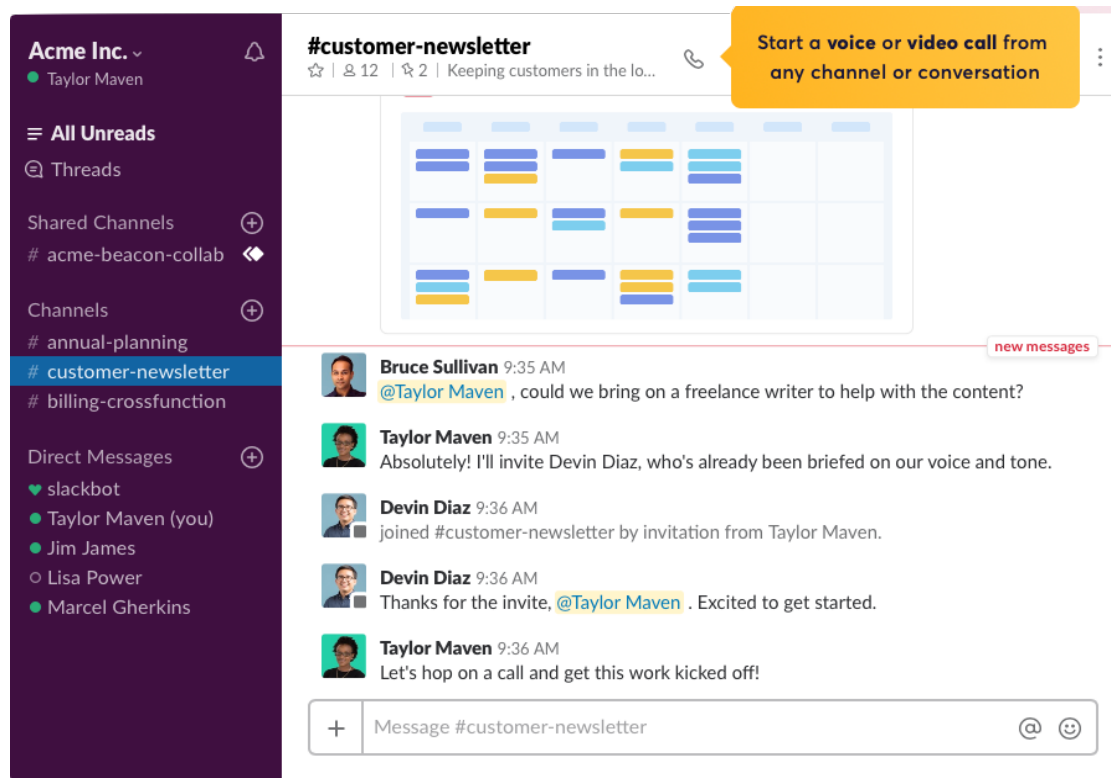
- Slack supports outside collaborators.



Managing and Tracking Communications

Example: Slack

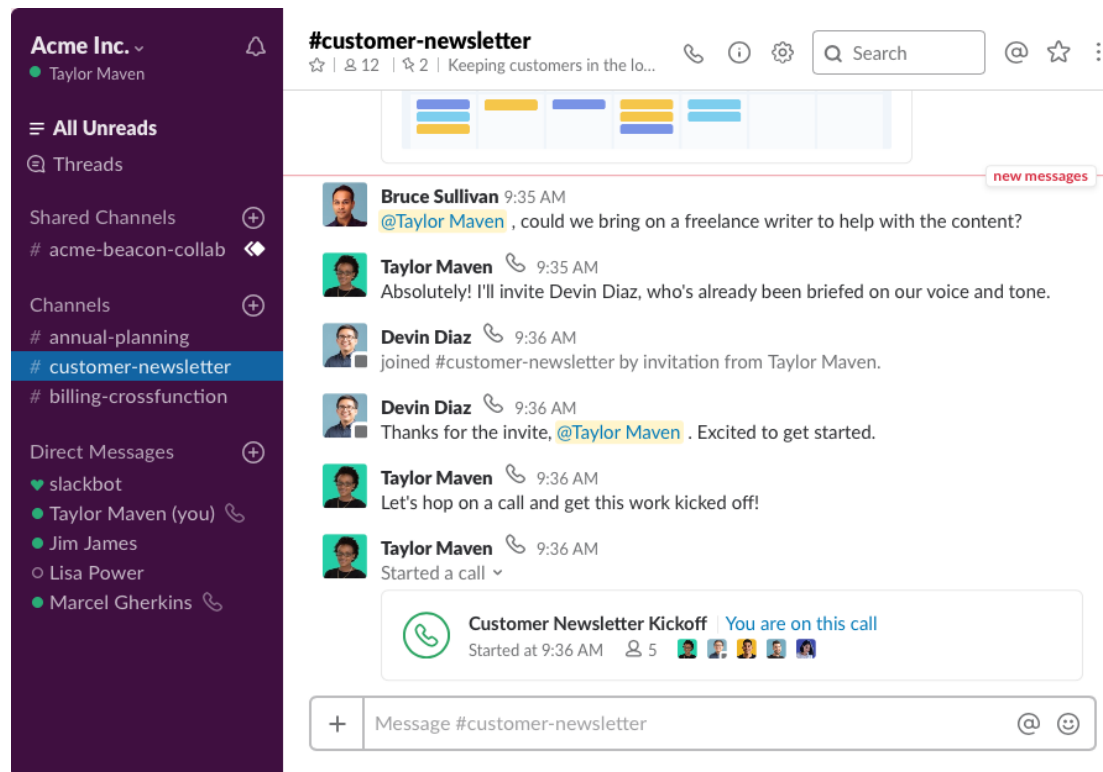
- Slack supports voice and video calls.



Managing and Tracking Communications

Example: Slack

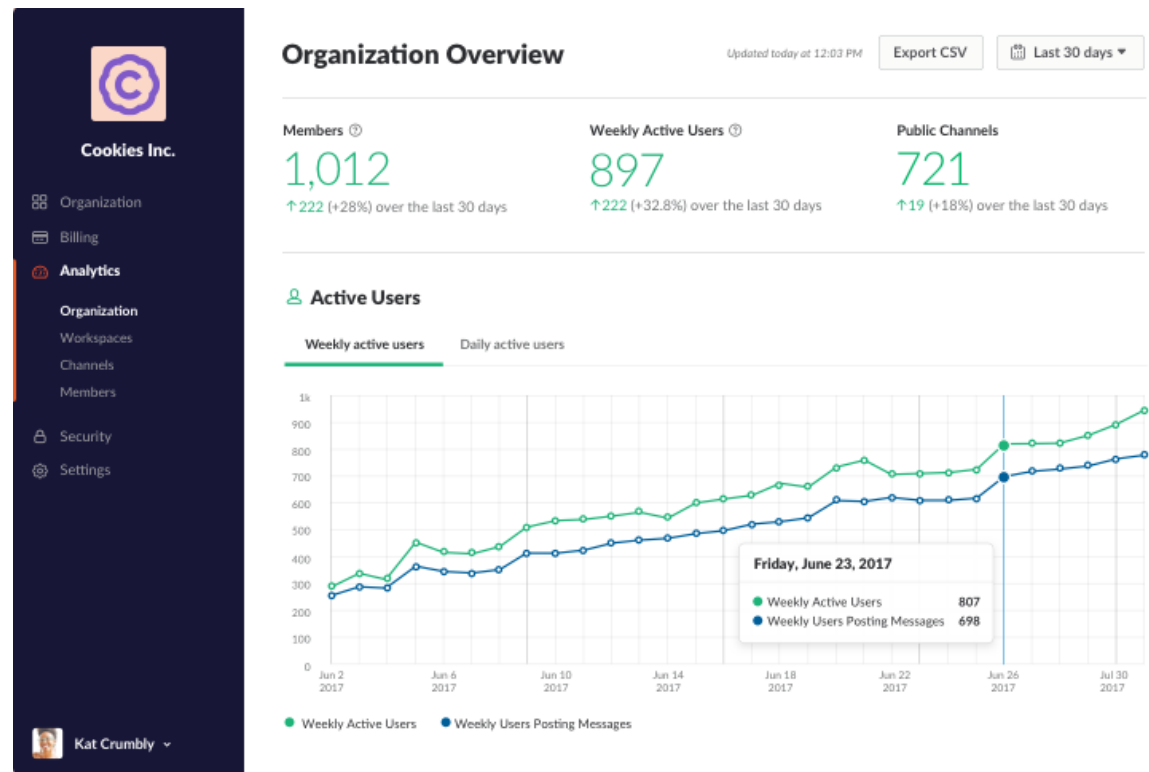
- Voice and video calls become part of the conversation.



Managing and Tracking Communications

Slack analytics dashboard

- Slack includes an analytics dashboard where teams can track communications and gain insight into how they use Slack.
- With an array of filters available, a team can sort and find information (e.g., messages sent in the last 30 days, or where conversations are happening).



Managing and Tracking Communications

Slack analytics dashboard

- Analytics overview
 - The analytics dashboard contains the information you need to understand how your workspace is being used.
 - See where people send the most messages and files (e.g. public/private channels vs. direct messages).
 - Organize public channels by metrics like name, total members, or messages sent.
 - See the total members on your workspace and the number of messages they've sent.

Managing and Tracking Communications

Slack analytics dashboard

- Active members
 - Weekly active members refer to the number of members (excluding bots) who read a public channel in the last 7 days.
 - Members who posted refer to the number of members (excluding bots) who posted at least one message in the last 7 days.
 - Daily active members refer to the number of members (excluding bots) with at least one cursor mark move in a public channel in the last day.
 - Daily members posting messages refer to the number of members who posted at least one message in the last day.

Managing and Tracking Communications

Slack analytics dashboard

- Messages and files
 - Only messages or files sent by members in channels or direct messages are counted. Bot messages, like the ones from a bot or a third-party app, are not included in analytics.
- Notes
 - Workspace analytics are updated once per day. It's not possible to manually update them.
 - Analytics are limited on the Free plan. Upgrading the plan unlocks other metrics and features.
 - All members except guests can view workspace analytics.

Managing and Tracking Communications

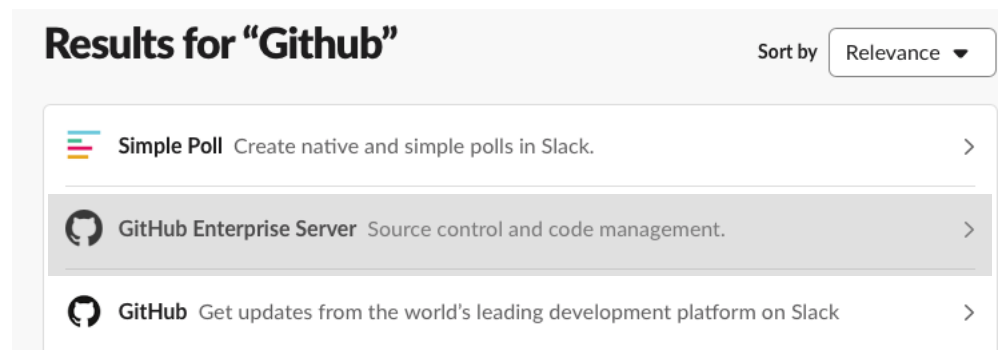
Slack apps

- Slack supports an *Apps* framework for adding new functionality. Slack includes many pre-built apps that provide integrations with other applications.
- A specialized type of app is a *Slack bot*. A Slack bot is an automation that enables user-defined code to be run when some condition occurs. We will introduce Slack bots later.
- First, we will look at one way to integrate GitHub Enterprise. The integration receives a notification from a GitHub server whenever contents of a repository are updated.
- When notification is received, the integration provides a SlackBot that sends a message to a specified channel.

Managing and Tracking Communications

Integrating Slack with GitHub Enterprise

- Integrating Slack and GitHub uses the Slack GitHub Enterprise Server app to capture GitHub events and report them as Slack messages.
- To install the app, select the **Add Applications** button on the workspace screen. In the search box, enter “GitHub” and select “GitHub Enterprise Server” from the results list.



Managing and Tracking Communications

Integrating Slack with GitHub Enterprise

- After selecting the GitHub Enterprise Server app from the list, the app presents a configuration editor for the integration settings.
- GitHub uses the generated *Webhook* URL to notify Slack of events. The URL is unique to the integration. Copy it now to configure GitHub later. We will not filter on any branches.

Webhook URL

When setting up this integration, this is the URL that you will paste into GitHub.

https://hooks.slack.com/services/T0106L30DUN/B0104M1TXB2/woKBssPCoA

[Regenerate](#)

Branch

Optional branch (or comma-separated branches) to filter on.

Managing and Tracking Communications

Integrating Slack with GitHub Enterprise

- The integration will post notifications to the “#software-project” channel. The descriptive label is displayed in the list of integrations, and the username is the sender of GitHub events.

Post to Channel
Notifications that come in from GitHub will be posted here.

software-project

[or create a new channel](#)

Descriptive Label
Use this label to provide extra context in your list of integrations (optional).

Lecture 10 Demo GitHub Integration

Customize Name
Choose the username that this integration will post as.

github


Managing and Tracking Communications

Integrating Slack with GitHub Enterprise

- The standard GitHub icon is used on messages from this integration and messages will appear like the sample shown. Select the **Save Integration** button to complete the configuration.


Customize Icon

Change the icon that is used for messages from this integration.

 or

Preview Message

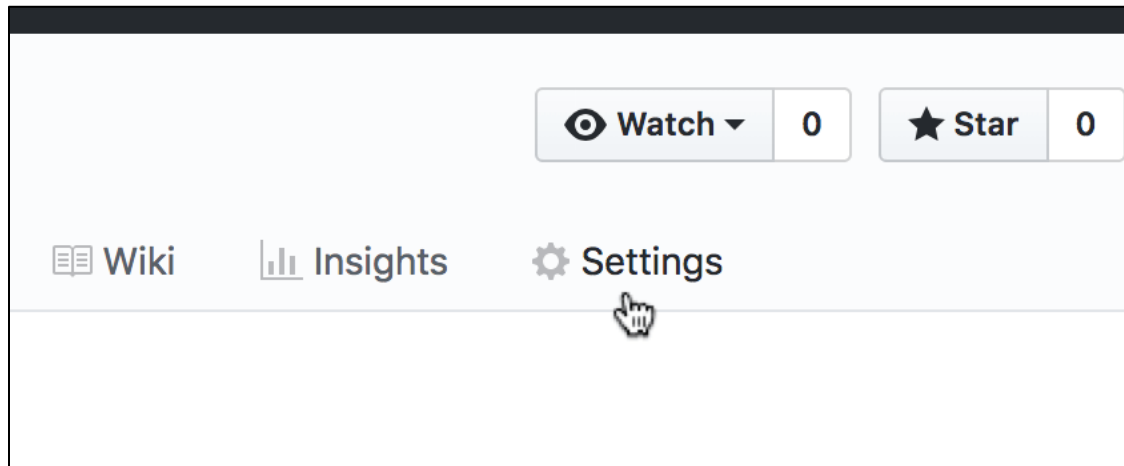
Here's what messages from this integration will look like in Slack.

**github** APP 10:59 PM
This is what messages from this service will look like in Slack.

Managing and Tracking Communications

Integrating Slack with GitHub Enterprise

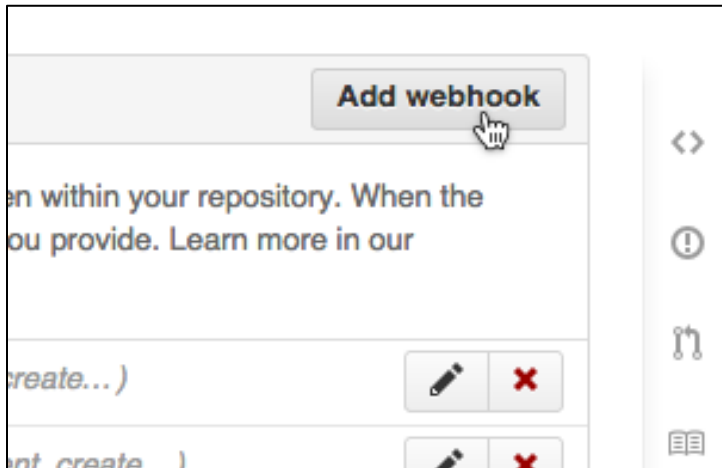
- The configuration dialog also includes instructions for adding a Webhook to your GitHub repository.
- **Step 1:** In your GitHub account, go to the repository whose events you want to monitor. Select the Settings tab in the top navigation.



Managing and Tracking Communications

Integrating Slack with GitHub Enterprise

- **Step 2:** Select on *Hooks* in the left navigation, and then select the *Add webhook* button



Managing and Tracking Communications

Integrating Slack with GitHub Enterprise

- **Step 3:** For the Payload URL, paste the generated URL provided by the Slack GitHub Enterprise Server configuration editor. The URL is unique to this integration.

Webhooks / **Add webhook**

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

`https://hooks.slack.com/services/T0106L30DUN/B0104M1TXB2/wc`

Managing and Tracking Communications

Integrating Slack with GitHub Enterprise


- **Step 4:** Ensure that *application/json* is set as the **Content type**, and that **SSL verification** is enabled.

Content type

application/json

Secret

SSL verification

 By default, we verify SSL certificates when delivering payloads.

☒ **Enable SSL verification** ☐ **Disable (not recommended)**

Managing and Tracking Communications

Integrating Slack with GitHub Enterprise

- **Step 5:** You can choose the events that you'd like to trigger notifications, but only commits, issues, and pull requests are currently supported. Select the **Add webhook** button when done.

Which events would you like to trigger this webhook?

☒ Just the push event.

☐ Send me **everything**.

☐ Let me select individual events.

☒ **Active**

We will deliver event details when this hook is triggered.

Add webhook

Managing and Tracking Communications

Integrating Slack with GitHub Enterprise

- The new webhook is now installed in the GitHub repository, and will be triggered when a *push* event occurs. GitHub sends a test payload to verify the Webhook.

Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

We will also send events from this repository to your [organization webhooks](#).

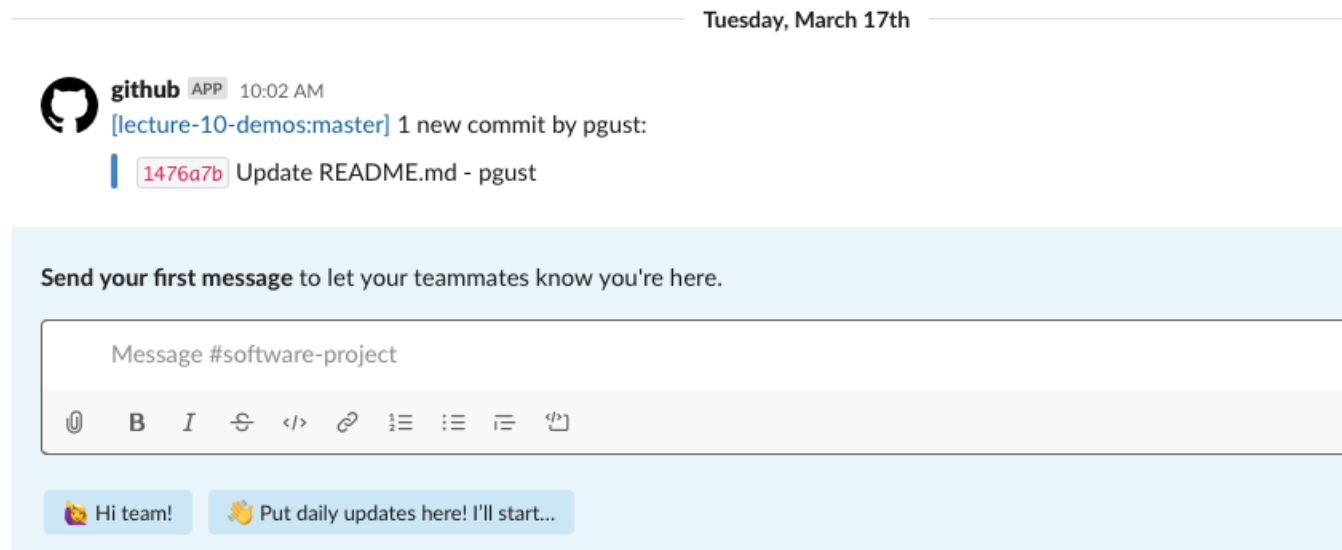
✓ <https://hooks.slack.com/services/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> (push)

EditDelete

Managing and Tracking Communications

Integrating Slack with GitHub Enterprise

- To test out the integration, we modify the README.md file in the “lecture-10-demos” GitHub repository and commit the change.
- As expected, the github SlackBot receives a notification from GitHub and posts a message to the “software-project” channel.



Managing and Tracking Communications

Slack bots

- A Slack bot is a type of Slack App designed to interact with users via conversation.
- A bot is the same as a regular app: it can access the same range of APIs and do all the things that a Slack App can do.
- When you build a bot for your Slack App, you are giving that app a face, a name, and a personality, and encouraging users to talk to it.
- Your bot can send DMs, it can be mentioned by users, it can post messages or upload files, and it can be invited to channels - or get kicked out.
- Bots are not cybernetic infiltration units. They are conversational interfaces for interacting with humans in Slack.

Managing and Tracking Communications

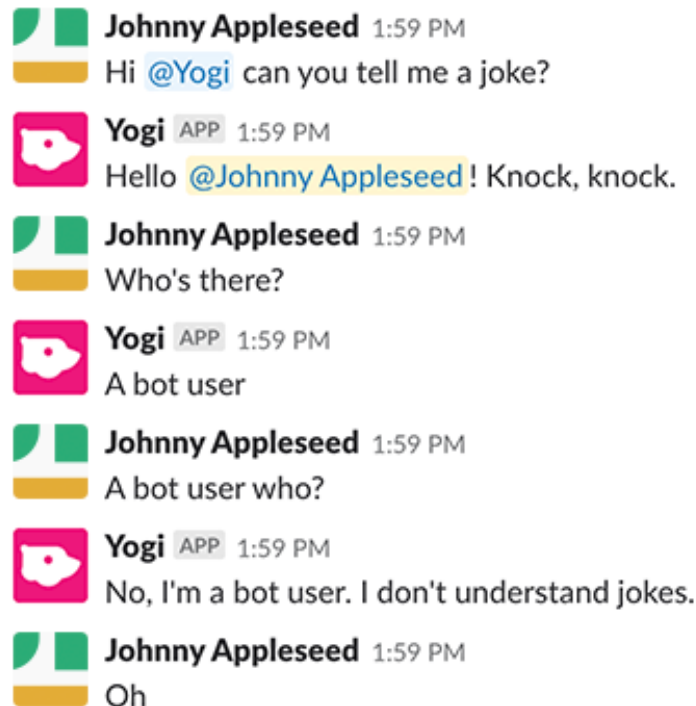
Slack bots

- Unlike a regular App, a Slack bot gets a **Bot User**, a **Display Name**, a **Default User Name**, and an **Online** flag to say whether the bot should always be shown online.
- Now the bot needs to be set up to handle certain events. Slack has a number of events a bot can respond to. For example,
 - ***app_mention*** is triggered when the bot name is mentioned.
 - ***message*** is triggered when a new message is posted
- Next, the bot must be installed into a Slack workspace, and set to a specific channel. Once completed, the bot will listen for user posts that mention the name of the bot.
- The final step is to tell the bot what to do when it hears something.

Managing and Tracking Communications

Slack bots

- Here is a dialog between a user named “Johnny Appleseed” and a conversational bot named “Yogi” that tells a “knock-knock” joke.



Managing and Tracking Communications

Slack bots

- There are 4 events triggered in this conversation:
 - the first is an `app_mention` event from the first message that mentions the bot
 - the next three are message events for each of the messages posted by Johnny Appleseed.
- Our bot will need to interpret each event and respond accordingly.
- The following example shows very simplified Express/Node.js router code of what your app logic might look like to implement the dialog we just saw.
- Do not worry if you are not familiar with Express/Node.js. the basic should be evident.

Managing and Tracking Communications

Slack bots

- Example “knock-knock joke” Slack bot router.

```
router.post("/", function(req, res, next) {  
  let payload = req.body;  
  res.sendStatus(200);  
  if (payload.event.type === "app_mention") {  
    if (payload.event.text.includes("tell me a joke")) {  
      // Make call to chat.postMessage using bot's token  
    }  
  } else if (payload.event.type === "message") {  
    let response_text;  
    if (payload.event.text.includes("Who's there?")) {  
      response_text = "A bot user";  
    }  
    if (payload.event.text.includes("Bot user who?")) {  
      response_text = "No, I'm a bot user. I don't understand jokes.";  
    }  
    if (response_text !== undefined) {  
      // Make call to chat.postMessage sending response_text using bot's token  
    }  
  }  
}
```

Managing and Tracking Communications

Slash commands

- The third way to integrate apps into Slack is through a *slash command*. A slash command allows users to invoke an app by typing a string into the composer box.
- We saw several slash commands used in the initial demo sequence for Slack. The first was
 - `/invite @DevinDiaz`
- This command is built-in to Slack and invokes functionality to send an invitation to Devin Diaz. The other was
 - `/salesforce MakerCorp`
- This slash command is part of the Salesforce integration app. It calls up the MakerCorp record from the team's Salesforce account and inserts it in the message.

Managing and Tracking Communications

Slash commands

- As another example:
 - */todo ask @crushermd to bake a birthday cake for @worf in #d-social*
- Here's the structure of the slash command:
 - */todo*
 - this is the command, the part that tells Slack to treat it as a Slash Command and where to route it.
 - *ask @crushermd to bake a birthday cake for @worf in #d-social*
 - this is the text portion; it includes everything after the first space following the command.
 - It is treated as a single parameter that is passed to the app that owns the command (we'll discuss this more below).

Managing and Tracking Communications

Slash commands

- A slash command app is created with the following values:
 - **Command** - the name of command, the actual string that users will type to trigger the command.
 - **Request URL** - the URL where the payload is sent when the command is invoked by a user.
 - **Short Description** - a short description of what the command does.
 - **Usage Hint** - displayed to users when they go to invoke the command.
 - **Escape channels, users, and links sent to your app** - turning this on modifies the parameters sent with a command by a user. Translates channel or user mentions into their correlated IDs.

Managing and Tracking Communications

Slash commands

- When a slash command is invoked, Slack sends an HTTP POST to the Request URL you specified. This request contains a data payload describing the source command and who invoked it.
- In the case of the /salesforce command shown earlier, the URL is to the team Salesforce server's HTTP server.
- The server handles the request by retrieving the MakerCorp record and returning it as the body of the HTTP response, in JSON format. This is interpreted by the slash app and inserted into the message.

Managing and Tracking Communications

Slack unified communications summary

- Unified Communication Systems (UCS) integrate information exchanged through multiple channels.
- Slack qualifies as a UCS because it
 - captures and tracks communications across channels
 - provides a common way to search for information across channels
 - archives communications for long-term storage and retrieval
 - offers data analytics that enable communications to be analyzed



Managing and Tracking Communications

Slack unified communications summary

- Slack's integration strategy
 - Allows information to exist in multiple silos but provides a common index that identifies places where the information is stored, enabling the communicated information to be located.
 - Gathers metadata for information stored across silos and enables it to be searched and presented in context.
 - Integrates the channels into a communication framework and provides access using a web interface and dedicated applications for commonly used platforms.

Managing and Tracking Communications

Slack unified communications summary

- Slack provides some functionality within the platform but relies on mechanisms to automate and integrate with external products.
- The primary mechanisms provided by Slack include
 - **Hooks** to receive events or communications from external products
 - **Integration points** for sending events or communications to external products
 - **Scripting** and other automation mechanisms for adding new behaviors that exchange information with external products