The Developer's Survival Guide to Email

BY **BRENT SLEEPER**

SPARKPOST

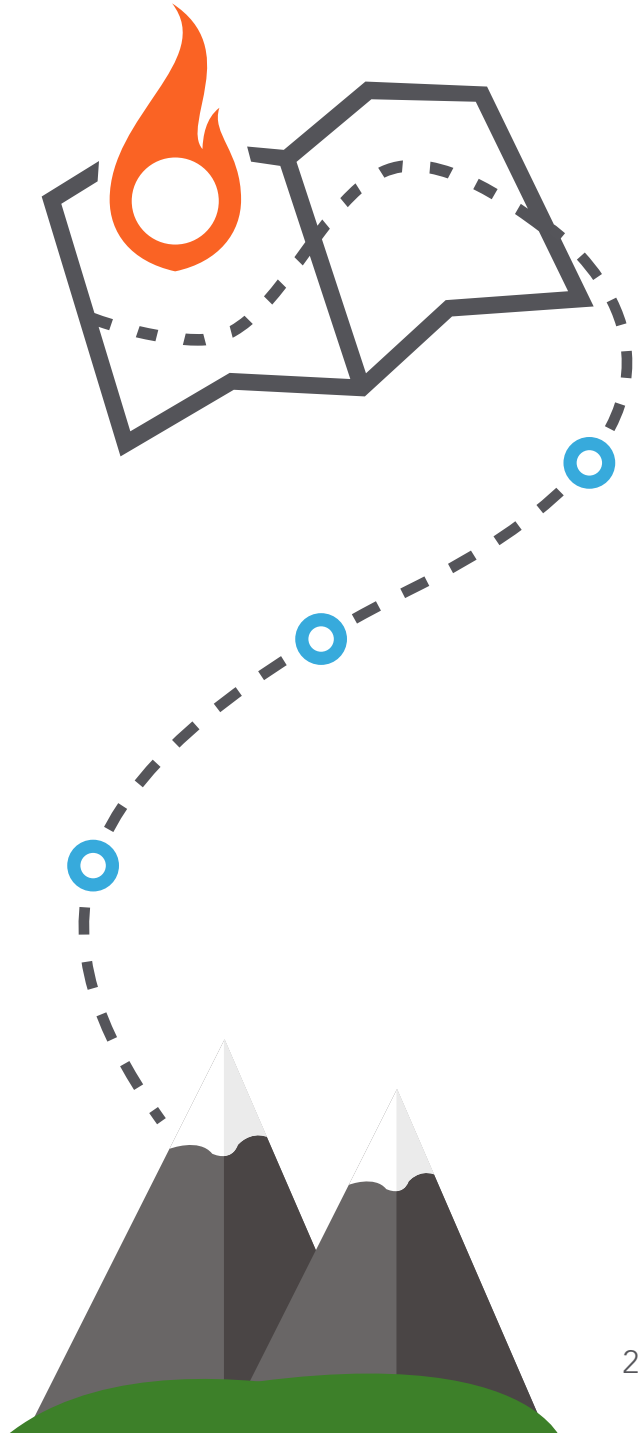
The Developer's Survival Guide to Email

"I Knew How to Validate an Email Address Until I Read the RFC." That's the title of a **10-year-old blog post**, but it neatly captures the exasperation that nearly every developer who works with email has felt at some point. There's really no getting around it: email can be a pretty strange beast with conventions, dependencies, and quirks unlike anything else, and many of us have a love/hate relationship with making it work.

We love email because it's ubiquitous, open, flexible, and effective; we hate—or perhaps have learned to treat it with a wary respect—because getting an email from point A to inbox B has so many variables outside our control. In short, it's easy to feel like "I Knew How Email Worked Until I Started to Build an App that Sends Email."

Fortunately, services like SparkPost let developers offload the operational considerations of building and hosting email infrastructure. We deal with the challenges of sending email at scale, the arcana of sender authentication, and the weeds of email deliverability so you don't have to.

But even so, if you're building email into your app or business process, email presents a number of messy details and idiosyncrasies that turn out to be "gotchas" for many developers. That's why we put together this guide: we're big believers in helping developers get the most out of email, and to be forewarned about email bugbears is to be forearmed. So here are 10 things you might not realize about implementing and sending email.



SURVIVAL TIP #1

Email at Scale is a Horse of an Entirely Different Color

One thing that surprises a lot of developers new to building email into apps and other systems is that email requires a very specialized skill set, and that email doesn't scale uniformly—or gracefully. With the right SMTP libraries and open source mail servers, it's relatively easy to send one email, or a few emails, or even hundreds of emails. So it's tempting to assume that once you've developed and tested the basic aspects of generating and transmitting a message, all you have to do is put your code into production, and you're done, right?

Not even close. At a pure infrastructure level, the very real challenges of resource management—the processing load, disk spooling, network I/O required to generate and send thousands or millions of messages at the kind of burst rates required for near-real-time message delivery—bogs down systems really quickly. And scaling to handle bounces and other deliverability signals from ISPs around the world is a whole other can of worms.

In short, email at scale is a highly specialized function that requires the expertise of experienced email development and operations teams to pull off well. That reality is one reason why email delivery services like SparkPost make so much sense. Our elastic cloud infrastructure won't even blink as it delivers as much message volume as any app or service can generate.

By the way, if you ever run into **our founder and CTO**, it's worth asking him about how he figured out how to optimize performance and resource requirements for message spooling back in the day by keeping it entirely in memory with zero disk I/O during message generation and transmission. It's pretty great.

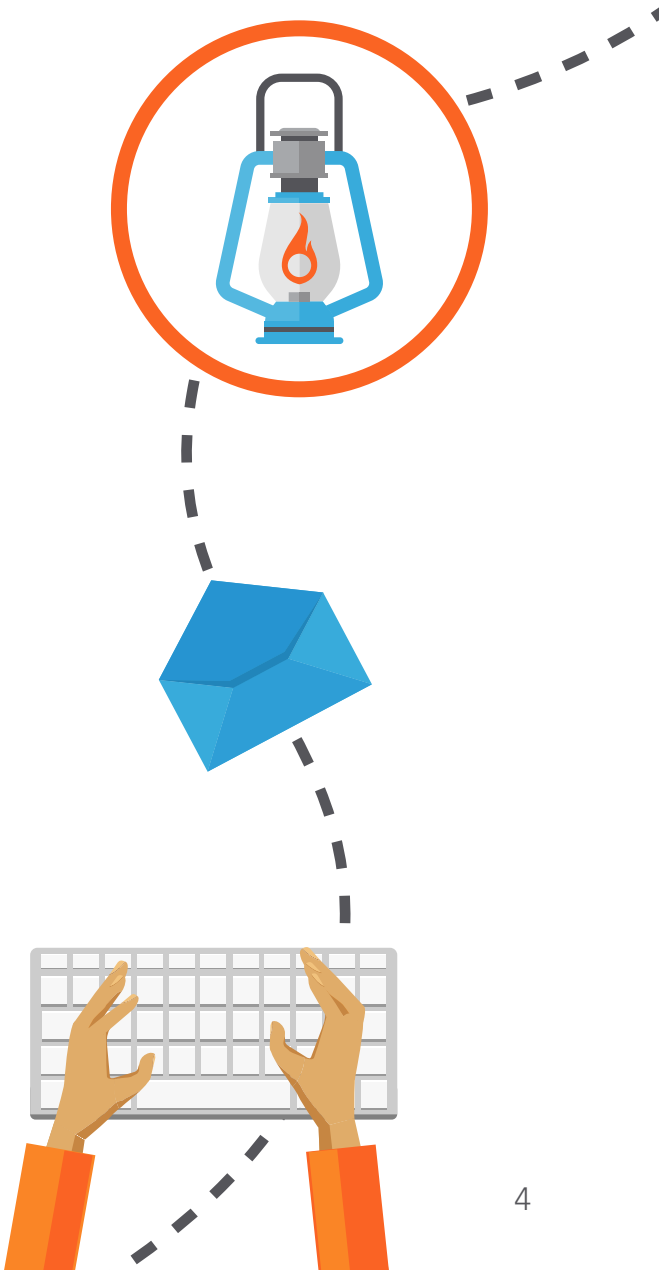
SURVIVAL TIP #2

Reaching the Inbox Can Be a Black Box

"Deliverability" is a fundamental metric for email senders, because there's no chance for a recipient to open, read, and respond to an offer if the email never arrives. The idea of deliverability seems straightforward. If I send 1000 emails, and my server's log files say 900 of those were accepted by receiving systems, then my deliverability is 90%. Simple, right? Yes… but no. That's because all your server knows is that the receiving system took the message, but not what was done with it. Did it go to the inbox? The spam folder? You don't know, because in both cases, the SMTP transaction is logged as a successful "250 OK"—SMTP doesn't differentiate spam from legitimate email.

To complicate the issue, every ISP has its own rules and policies for determining which messages to accept, which to reject outright, and which to put into a sort of gray zone of heuristic evaluation. Because so many spammers and other bad actors try to game the system, these criteria are kept highly secret by the ISPs.

Services like SparkPost give deliverability a lot of attention, and our ISP relationships and the intelligent automation of our **Adaptive Email Network** make a big difference. But, understanding that different ISPs might treat identical emails differently (maybe Gmail puts it in the inbox, while Yahoo flags it as spam, for example)—and what you can do about it—is sometimes challenging. If you want to learn more about improving your email deliverability, we've created a number of **helpful guides all about it**.

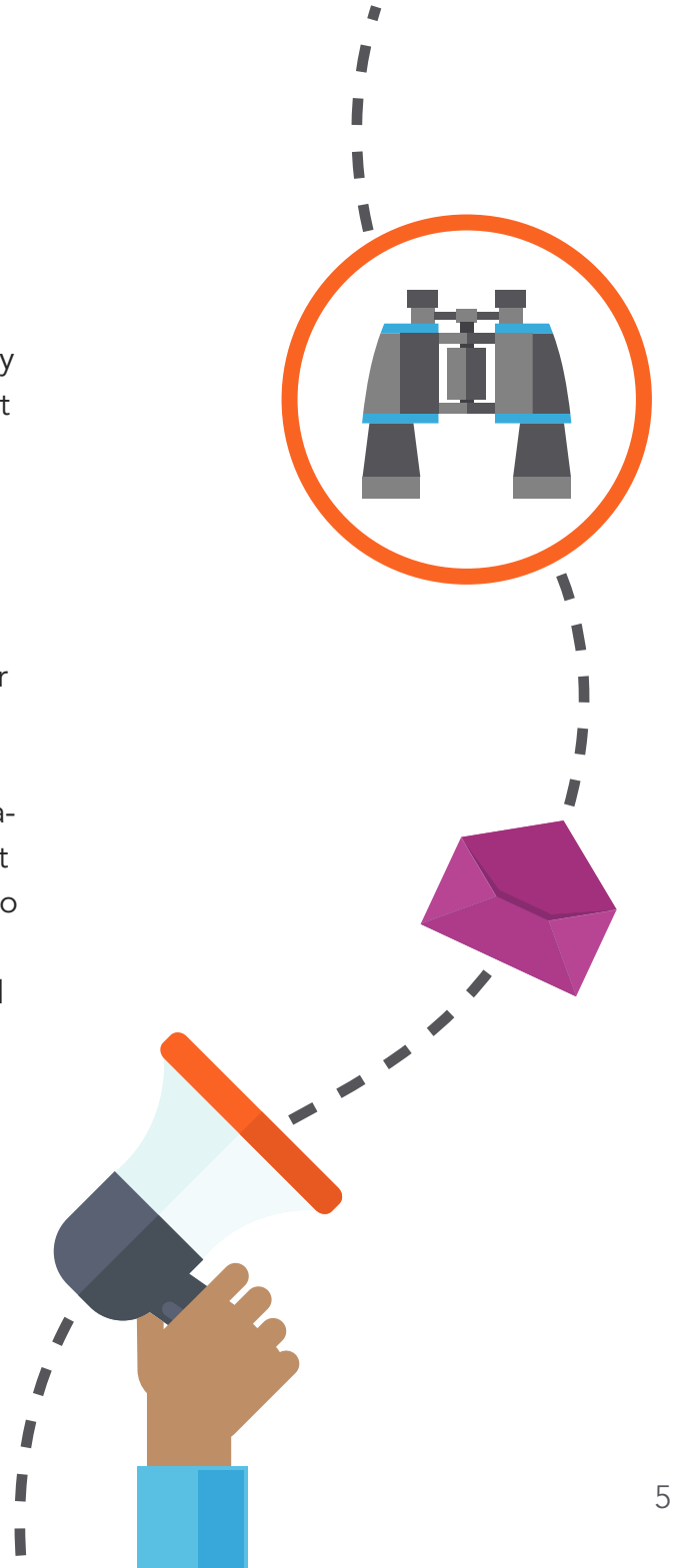


SURVIVAL TIP #3

Your Email Has a Reputation

Not caring about her reputation made a great proto-punk **hit song for Joan Jett**. But, believe it or not, email senders (and their associated IP addresses) actually have reputations, too, and ISPs take them very seriously. In fact, sender reputation is one of the most important concepts in message deliverability. Every IP address and domain has a reputation based on factors such as past rates of bounces and spam reports. Domains and IP addresses with a good reputation have a better delivery rate, so building and maintaining a good reputation is central to the mission of email operations.

SparkPost's operations and deliverability teams are assiduous in making sure that our shared IP addresses have great reputations, and that's one reason for the **messaging policy** in our terms of service. But as someone building an app or system that uses email, you can do a lot to affect the reputation of your own domain. **SparkPost's guides to improving email deliverability** include great advice for managing email reputation, including: making sure the content you send is welcomed by recipients who actually asked for it, taking technical measures like email authentication, and following best practices for the rate of message sending, and more.



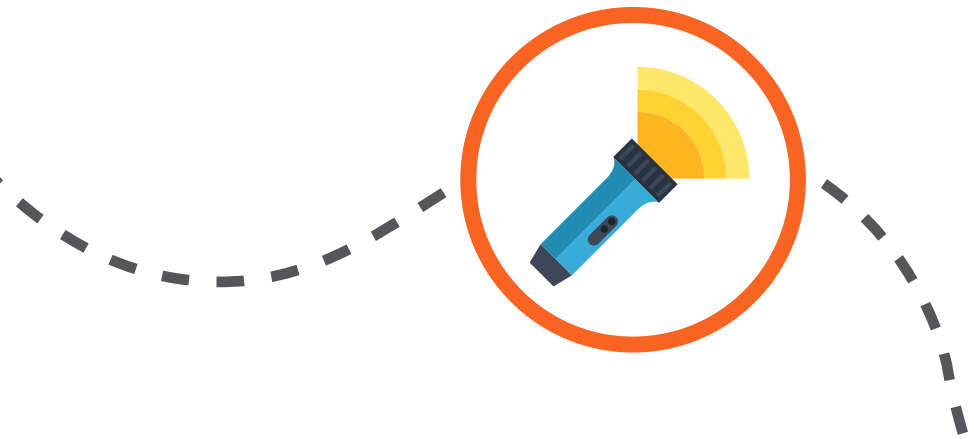
SURVIVAL TIP #4

Doing the Right Thing with Email Isn't Just a Good Idea—It's the Law

If there's one thing that unites all of us that deal with email, it's spam. You hate spam. We hate spam. Everyone (except spammers) hates spam.

But what surprises some developers new to email is that there are some very specific guidelines that govern email that gets sent to customers. Some of these are considered best practices that keep customers happy and give you great results. But other aspects actually are enforced by law.

In the U.S., there's a law called "**CAN-SPAM**" that requires all senders of commercial email enable recipients to opt out. In Canada, the "**CASL**" law has stricter rules (and governs a number of other rules about email). The European Union and other countries have strict regulations as well.



The details of what each of these laws cover are too specific to go into here. (Dammit, Jim, we're developers, not lawyers!) But a basic thing that anyone sending email for a business needs to know is that:

- → You should only send commercial email (that means "marketing" email, very broadly defined) to users who specifically told you they want to receive it. (In email parlance, that means users who have "opted in.")
- → Specific types of emails like receipts that are a necessary part of doing business with a customer are called "transactional email" and get certain exemptions from some of these rules.
- → SparkPost's terms of use policies require compliance with rules like these, and also enforce several other best practices that help ensure every email that our customers send is a good email, not spam.

The **SparkPost web site** and **blog** have good information about CAN-SPAM, CASL, and other email rules and best practices.

SURVIVAL TIP #5

HTML Like It's 1999

Email recipients today expect (and respond to) emails that look like the content on the web and in mobile apps; there's simply no question that **HTML rules the inbox**. But what surprises just about everyone who begins to develop for email is just how idiosyncratic HTML support remains in many email clients and webmail inboxes.

Although some email clients (notably Apple Mail, including on the iPhone) are good to go for HTML5, others (like Outlook) are downright terrible in their support for modern HTML standards. Moreover, webmail inboxes like Gmail each have their own quirks and limitations.

As a result, email developers need to dust off the 1999 edition of your favorite reference guide and code to a subset of HTML 4.

Notable limitations of coding HTML email include:

- A reliance on tables for layout
- The need for inline, redundant CSS
- No Z-layering or dynamic rendering of <divs>
- Unpredictable display of background images
- Limited support for web fonts
- Little or no support for JavaScript, video, animation, forms, and other types of interactive content

Fortunately, there are ways to make developing email HTML a little less painful. A quick **Google search** will turn up dozens of practical guides to using HTML in email, and one of our own developers recently made life better with **automatic CSS inlining in SparkPost templates**.

SURVIVAL TIP #6

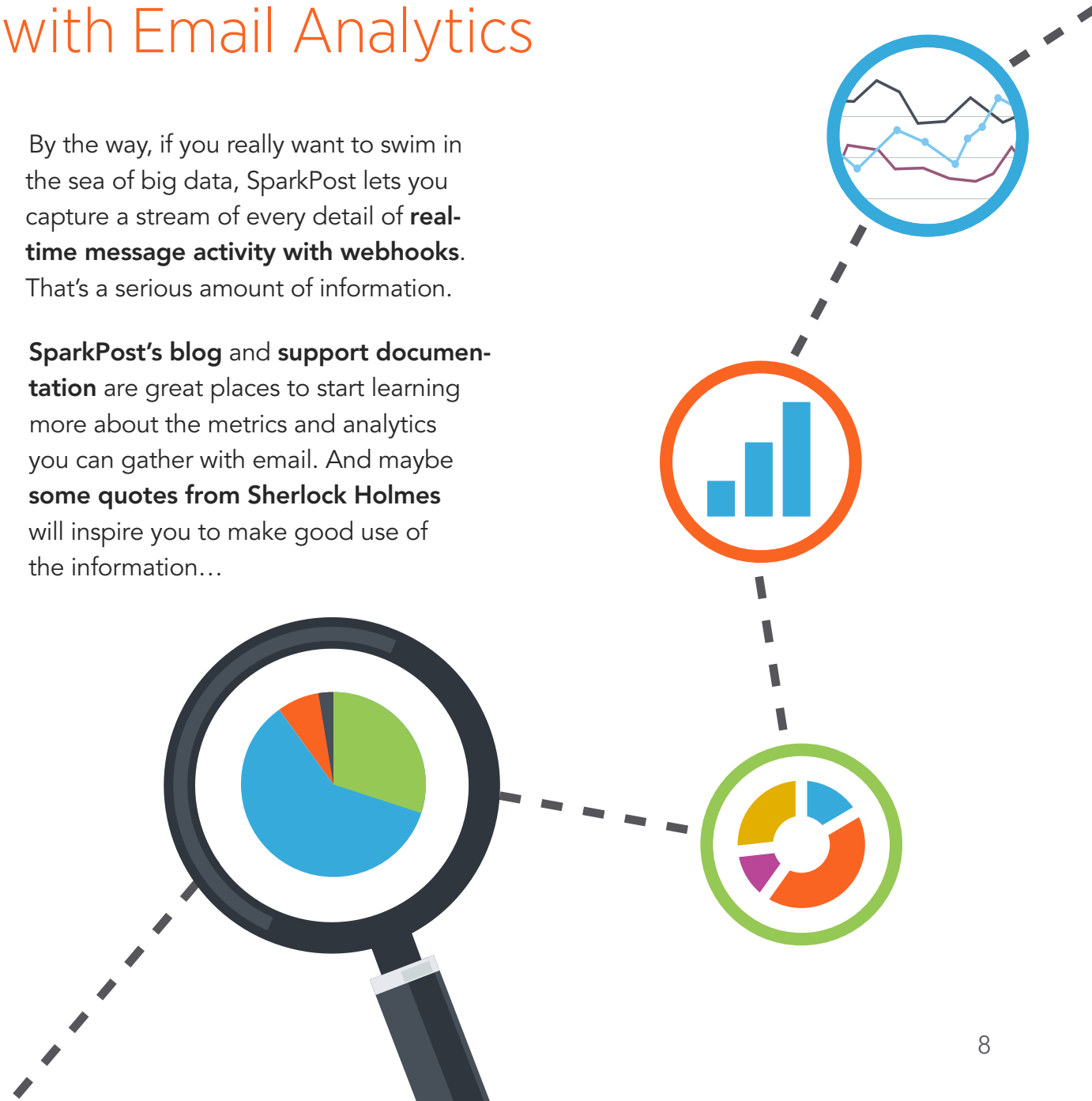
Connect the Dots with Email Analytics

One of the great things about email is that it comes with a lot of useful data. But because email isn't a closed system, drawing conclusions from the data requires a solid understanding of the metrics available, good instrumentation, and deductive reasoning. (For instance, **measuring whether an email reached the inbox** must be approached from a couple different angles to get the full picture.)

SparkPost helps get you there with great instrumentation of our service and messages, and we give you the kind of tools you need to query it. Our real-time analytics metrics dashboard reports over 35 different metrics, and you can customize the reporting interface to drill down as you need.

By the way, if you really want to swim in the sea of big data, SparkPost lets you capture a stream of every detail of **real-time message activity with webhooks**. That's a serious amount of information.

SparkPost's blog and **support documentation** are great places to start learning more about the metrics and analytics you can gather with email. And maybe **some quotes from Sherlock Holmes** will inspire you to make good use of the information…



SURVIVAL TIP #7

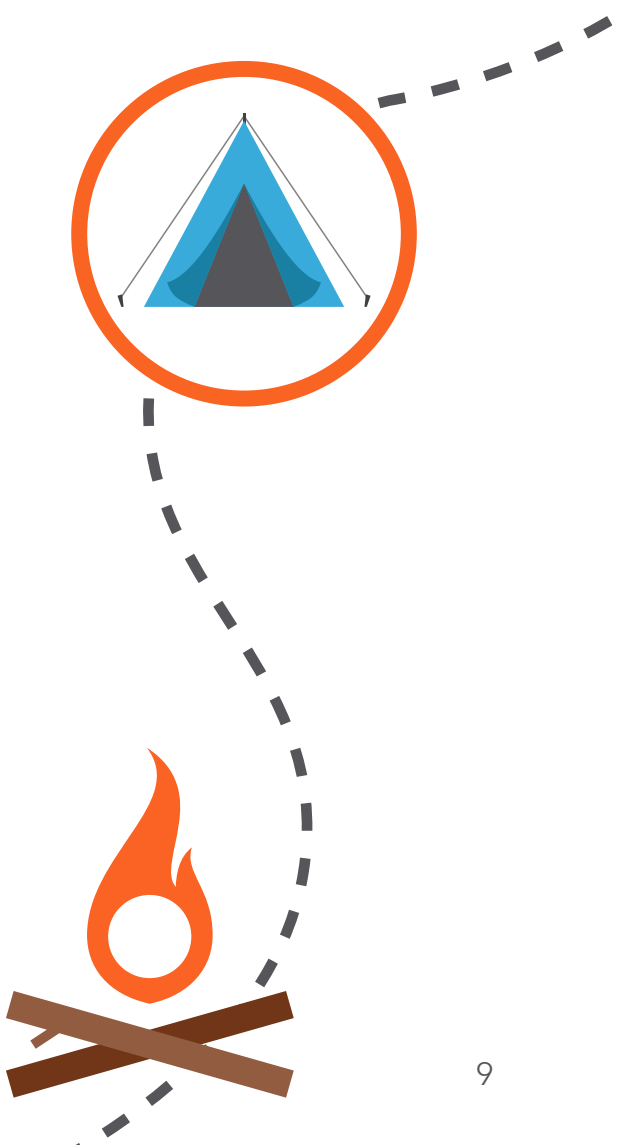
Email Depends upon the Kindness of Strangers
(Or At Least the Scaffolding of Other Infrastructure)

Email as a concept predates the Internet, and the implementation we use today was **created in the very earliest days of the Internet**. That history has two particular ramifications. First, its architecture relies very heavily on other core Internet infrastructure like DNS. Second, many of the things we take for granted as necessary in modern systems (like security) were bolted onto email over the years in ways that are simultaneously pragmatic and reflect the reality of design-by-committee and diverse vendor and user interests.

In short, some parts of email infrastructure need a lot of fiddling with to make work. Depending upon the nature of your organization, it also might require the cooperation of other teams like systems administrators. Here are some basic things you'll need to get configured to ensure your email gets handled in the best way possible:

- → Basic DNS configuration, including reverse DNS and MX records, for sending and bounce domains
- → Additional DNS configuration with special TXT records to support authentication standards like SPF, DKIM, and DMARC
- → Configuring your SMTP host with administrative role accounts like postmaster@ and abuse@

The SparkPost support site has concise how-to's for **verifying sending domains**, and our blog is a great resource for practical advice for working with standards like **DKIM** and **DMARC** (and why, even with its complexity, **email authentication really matters**).



SURVIVAL TIP #8

Email Is Not One Size Fits All

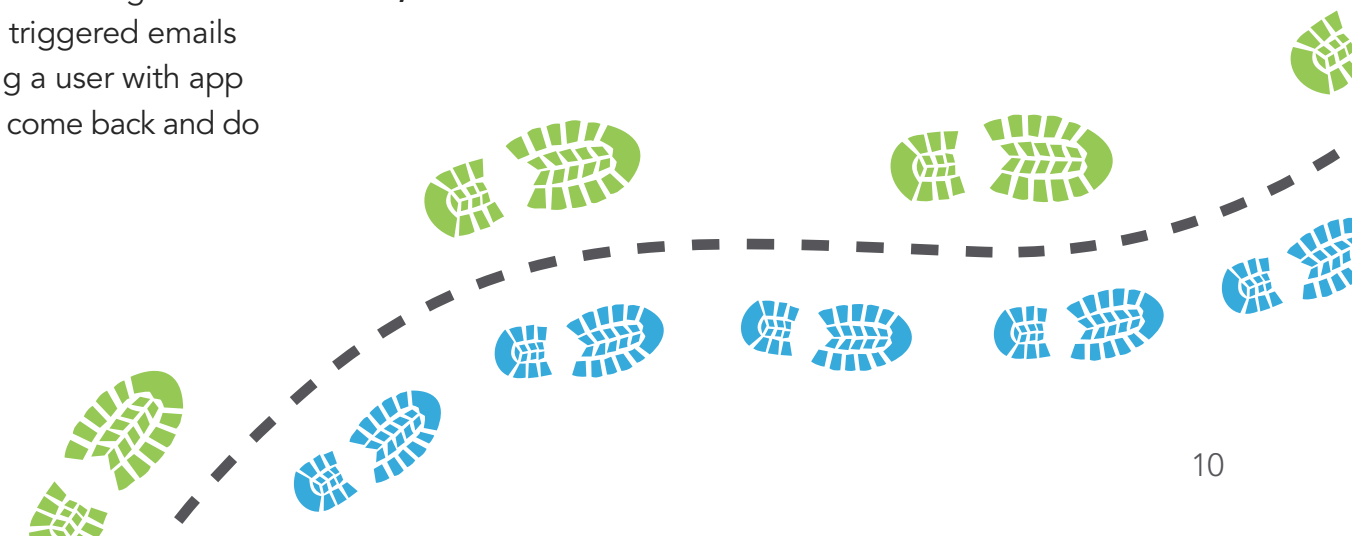
Think about what you have in your own email inbox: Personal messages from co-workers or friends, sure. But what else? Newsletters? Marketing promotions? Receipts? Password resets? Welcome emails? We've all received all of the above, and more.

One of the great things about email is that it's totally flexible. No matter what you're building, you can adapt email messaging to communicate with your users. But each of these use cases needs a different kind of email: a different template, sure, but also a different set of practices for how you trigger, generate, and send them.

- **Triggered email** is what developers using SparkPost most often are looking to build: messages generated in real time based on an activity or data-based trigger. Sometimes triggered emails are "transactional email"—something like a receipt or a password reset. Other times, triggered emails play more of a marketing role, like helping a user with app onboarding steps or encouraging them to come back and do something on your web site.
- **Bulk email** is what a lot of us think of as "marketing email." Newsletters, sales announcements from a retailer, and so on. When done right, your customers are happy to receive these. When done wrong, they come across as unwanted spam. Don't be that kind of sender. Make sure your customers opted in, and make sure you're giving them information they actually requested.

Understanding the differences between these types of email is a really big topic, and it affects everything from the design of your emails to rules about how and when they are sent. (See survival tip #4 above for really important information about these rules.)

If you want to learn more about different types of email, check out **SparkPost's guides to transactional email, triggered email, and more**.



SURVIVAL TIP #9

Email Is Mobile, and Mobile Is Email

Email's been around a really long time. So long, in fact, that "the death of email" has (incorrectly) been proclaimed over and over. With the ubiquity of smart phones, it's an easy leap to think that social media, text messaging, or apps have made email on mobile devices obsolete. Of course, those things are really important forms of connecting with users. But the (perhaps surprising) truth is that not only do we send and receive more email than ever, but a majority of email is read on iPhones, Androids, and the like. Litmus, a company that tracks data like this published some **striking figures about email client market share:**

- → 54% of email is opened on mobile
- → 26% is read in a webmail tool like Gmail
- → 20% is read in a traditional, desktop email client

So there's no question that email is genuinely cross-platform, and that mobile is a critical piece of that. That's a big reason why SparkPost offers enterprise developers **one platform and set of APIs to generate email, SMS, and push notifications**.

SURVIVAL TIP #10

Make the Most of SparkPost Developer Resources

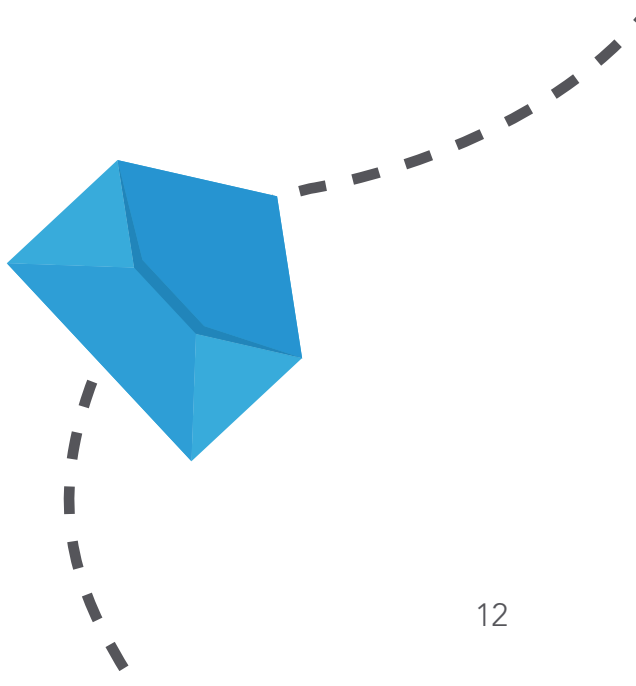
At SparkPost, #WeLoveDevelopers. Whether it's a simple web app or a complex enterprise business process, we make it easy for developers at companies of all sizes to build email into whatever you're coding.

Want to learn more? Here are some great developer resources to bookmark:

- **The SparkPost Developer Hub** is a collection of resources to help you succeed with SparkPost, including details about our API, client libraries, and more.
- **The SparkPost community Slack channel** is a great way to learn from other developers using SparkPost—and SparkPost's own team as well.
- **The SparkPost support site** has all the docs and technical support you need to make the most of our product.
- **The SparkPost blog** regularly features email tips, developer how-tos, and sneak peeks at new product functionality.
- SparkPost is on Twitter at **@SparkPost** and **@SparkPostDev**.

But the truth is, there's no better way to learn than by trying.

Try SparkPost for FREE today!



SPARKPOST

SparkPost.com • 301 Howard St., Suite 1330 • San Francisco, CA 94105 • tel +1 415-578-5222 • toll free usa 877-887-3031