RANDALL DEGGES (/)

Two Years of Evangelism

In a couple short months, I'll have been at Stormpath (https://stormpath.com/) for two full years. It's pretty insane how fast the time has flown by. It seems like only yesterday I was writing

(https://www.rdegges.com/2014/moving-on/) about my decision to move to Stormpath, and leaving my startup behind.

Since I'm finally taking a bit of a break to sit back and reflect about everything that's happened over the past two years, I figured now would be a good time so share some of the things I've learned and experienced along the way.

Developer Evangelism is Kinda Crazy

First off, being a Developer Evangelist is sorta crazy.

Before joining Stormpath, I never held any sort of marketing / sales / publicity related positions *anywhere*. Quite the opposite, actually – I was just a developer, working out of my home office, spending 99% of my time completely isolated from the world hacking on code =)

When I started at Stormpath, I was fairly worried about this!

I've been to enough events (*meetups, hackathons, etc.*) in my life to have met a lot of other Developer Evangelists, and I know secondhand that a big part of being a *"Developer Evangelist"* is doing lots of public stuff: giving tech talks, meeting people, breaking the ice, etc.

While I'm good enough in one-on-one conversation, I'm truly an introvert at heart, and socializing really takes a lot out of me. To say I was worried about this is a bit of an understatement. When I accepted the job offer I was incredibly nervous!

In addition to spending a lot of time actually talking with people, I knew I'd have to give at *least* a few tech talks – and I know from previous experience that I have a very hard time getting up in front of an audience of people and clearly conveying a lesson in a fun and interesting way.

Not only does public speaking make me really nervous, but it also takes a ton of prep time and practice in order to go smoothly.

I guess what I'm trying to say is: I knew going in that I was signing up for something that felt very uncomfortable to me, but in the end, I'm glad I said *"fuck it"*, and took the leap, as I really wanted to challenge myself to do something new, and push myself into uncomfortable territory.

What Developer Evangelists Do

There are several different ways to be a successful Developer Evangelist (*I intend to write more about this in the near future*):

1. You can attend lots of events (*meetups*, *hackathons*, *conferences*), and do a lot of small group communication.
2. You can write a lot of open source libraries and tools that developers use, and spend a good portion of your time maintaining them.
3. You can spend time working on your company website and documentation, improving the developer experience directly at the source.
4. You can build a lot of sample apps that use your product or service. These serve as examples for developers using your stuff of how to do things properly.
5. You can write a lot of content: articles, educational material, etc. This is a great way to educate your audience, and build a fan base.
6. You can do a combination of the above.

What I Do

My first few months at Stormpath, I invested myself heavily in strategy #1 – mainly because this is what I've seen other people doing.

Many of my Developer Evangelist friends spend about 90% of their time traveling, attending events, and giving talks.

While this is a tried and proven strategy for reaching developers with your product / service, it didn't really feel *right* to me.

Instead, after some experimentation, I've fallen into the habit of doing a lot less of strategy #1, and a lot more of strategy #2. I now spend most of my time directly building and improving the Stormpath client libraries in a variety of languages: Python, Flask, Django, Node.js, and Express.js.

Despite my initial uncertainty about this, user feedback has shown me that spending a lot of time focusing on the quality of individual library implementations is a great way to on-board users, improve user experience, and generally attract new users to the service.

Since Stormpath (https://stormpath.com/) is a fairly complex API service (*it stores user accounts, user profile data, and handles all sorts of web authentication stuff: login, API auth, single sign on, etc.*), our client libraries are incredibly important – if they aren't easy to use, simple, and intuitive – developers will immediately move on to something else.

Spending time building and improving these libraries has been a great way to not only popularize the Stormpath service – but has also given myself (*and everyone at Stormpath*) a much more in-depth view of what developers actually want out of our service, and what all of us should focus on building.

When I'm not doing stuff like library maintenance, I'm typically splitting my time up between:

- Authoring blog content. Writing technical articles are how to do X or Y – typically security related topics. These tend to raise our organic search traffic quite a bit, and serve as a nice funnel for marketing: people searching for how to do X or Y learn what they need, and also learn that we exist! Yey!

- Giving talks at local meetups or small conferences. Giving talks has turned out to be a great way to both get your service noticed, as well as make friends with people who are actually interested in what you're doing. After giving a talk, I typically end up chatting with at least a few people who are deeply interested in the talk topic, and subsequently, my work.

- Working directly with customers, helping them get stuff done. Through support requests, I end up doing a lot of pair programming with customers. This is actually one of my favorite parts of the job. I get on a screen sharing session and hack on a customer's code base for a few hours, solving

problems and getting things done. This makes people really happy, is very fun, and makes me feel like I'm having a direct positive effect on someone.

What I Don't Do

As important as knowing what you *need to do* is, there's also a lot of value in knowing what you *shouldn't be doing*.

I alluded to this earlier, but in my first few months at Stormpath, I took a somewhat typical evangelism approach to the product:

- I went to a lot of hackathons.
- I went to a lot of conferences / events.
- I tried to do as much *public* stuff as possible.

Almost immediately, I realized this was just not a good idea. Here's why:

Unless the product or service you're evangelising is already very far along, well polished, etc., there's no point in spending a lot of your time doing in person things: hackathons, events, conferences.

The reason why is this: unless you can very clearly show developers what your service does, and how to use it in an incredibly simple, fool-proof way: you're just wasting your time.

When I first joined Stormpath, we had essentially no developer libraries, so using the product was incredibly difficult. A developer who wanted to store user accounts in Stormpath, for instance, would have to learn about our low-level client objects, dig through Github source code, and do some heavy hacking to make even the simplest of integrations.

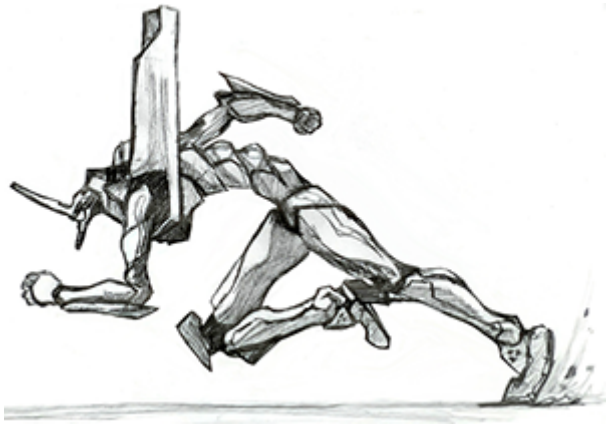
From a developer perspective: it was simply not worth it.

In the first few months I learned the hard way that I couldn't just copy Twilio's (https://www.twilio.com/) evangelism strategy – I'd have to make my own.

Even today, almost two years later, we're still not at the point where we can confidently go out into the community at hackathons and various events and easily show off our product in a variety of languages. We still have a ways to go in terms of integration support, ease-of-use, and overall simplicity.

But we're making great progress =)

What I'm Moving Towards



I think that in the long run, what would work best is to split my time up as follows:

- Spend 3 days every week working on library development and maintenance work. This is the most important part of the job in my opinion, as it is what directly effects developers the most. If you're evangelising an API service, for instance, your product is only as good as your weakest developer library.

- Spend 1 day per week working on content related stuff: blog posts, screencasts, tech talks, etc. This is where you reach the most people: educational materials. At Stormpath, almost all of our growth and success has been due to a combination of building high-quality developer libraries, and writing lots of very specific technical articles that people find useful.

- Spend 1 day per week doing assorted other things: meetings, planning, collaborating on projects with other team members, helping with the public website / on-boarding, etc. At Stormpath, for instance, I frequently get involved in various other areas not directly related to evangelism: company analytics, on-site hackery with larger companies, working on the public facing website, giving feedback to other teams, etc. All of these things are valuable, and instead of being completely *siloed* into *evangelism*, it's important that you also make time

for helping in other areas that you can contribute to as well. This helps to build a good *team culture*, and also keeps you from falling into the same old routine day-after-day.

Over the next year, I'm hoping to slowly transition more and more into something like the above, as the last two years have shown that the three above items are the most effective for Stormpath at this time.

The Main Problem: Lack of Time

The main problem I face on a day-to-day basis is, quite simply, lack of time.

One of the biggest problems I've had through my first two years (*and still a bit currently*) is deciding what the priority of various tasks are. It's very difficult to prioritize large, high value projects when there's a constant influx of important time-demanding things going on: responding to customer requests, fixing library bugs, reviewing pull requests on Github projects, etc.

Sometimes, at the end of a busy day, I'll have spent 10 hours working on lots of important customer related tasks, but have made no progress on my first (*and sometimes only*) task for the week.

This can definitely be a bit demotivating, but I've learned to aim for small wins here and there. For instance, instead of putting a huge project onto my task list, I'll break it up into 10 separate mini-projects. This way I can knock one or two of these mini-projects off my list every day and be able to sleep peacefully at night knowing I've accomplished something.

Another thing I've learned (*just recently!*) is to be ruthless with your TODO list. I'm currently using Trello (https://trello.com/) to manage my backlog of tasks, what I'm currently working on, and my weekly *"Done"* list. This has been great because it gives my coworkers insight into what I'm currently busy with, and also gives both them and me a way to co-ordinate about what things need to get finished first, etc.

Whatever system you end up using to manage your time, make sure you use it religiously and don't deviate.

Never, Ever, Stress Yourself Out

I realize I just wrote for quite a while about how crazy being a Developer Evangelist can be, but as counter intuitive as this might seem – no matter how crazy things get, never allow yourself to stress about work.

It's just not worth it!

At the end of the day, there are always a million things to do:

- Projects to build.
- Articles to write.
- Emails to send.
- Weights to be lifted.

But the reality is that this is all just part of *your life*. This is your life that you're living! You are the one who makes decisions about what to do with your time, and at the end of the day you're the one who has to live with those decisions.

Instead of constantly thinking about the past or future, and worrying yourself with a seemingly ever-increasing list of chores, simply focus on one thing at a time and enjoy your life as much as possible.

I know from past experience that it's really easy to let yourself get wrapped up in the scarcity mentality: no time for this, too many things to do for that, etc! If you're a Developer Evangelist, there are *always* new things you're learning and working on, so it's *absolutely, 100% necessary* to just relax, and focus on one thing at a time.

Public Speaking is Very Hard

Another thing I've come to really appreciate is how difficult public speaking really is.

I mentioned earlier that I'm a bit introverted and afraid of public speaking – but that's not quite true. I know that if I practice enough and spend sufficient time in preparation, I'm able to give a relatively solid talk and feel confident in my performance.

What's hard about public speaking is developing a clear and concise talk that's both technical *and* entertaining at the same time.

I tend to watch a lot of technical talks for fun. This is partly because I enjoy going to meetup groups, and partly because I often stumble upon highly recommended ones on Youtube and such. But the point is: I watch a lot of them.

One of the things that has always sorta bugged me is that many of the best technical talks out there are… Very, very dry. To put it bluntly: they're boring. SUPER BORING. So incredibly boring that it's honestly hard to follow.

When I gave my very first ever tech talk years ago, I decided that above all else I want to be the sort of speaker that doesn't bore his audience.

Over the years I've experimented with many approaches, but what I've found works best is raw enthusiasm.

It doesn't matter whether I'm speaking about password security or scaling API services, I try to embed as much passion and enthusiasm for my topic into my talk as humanly possible.

But here's the tricky part: it's very hard to write good technical talks that are:

- Sufficiently easy to follow.

- Sufficiently technical so that I can provide practical, in-depth information.
- Sufficiently interesting so the audience isn't bored.
- Sufficiently complex so that I'm able to teach everyone in the room something new.

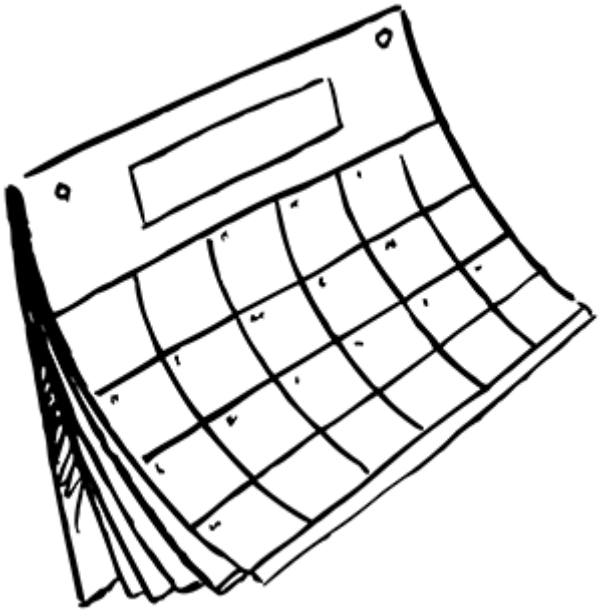
It's very hard to balance all four points.

I'm still working on this quite a lot, and with each new talk I write I try to improve at least a little bit more towards my end goal.

But damn, writing a good talk is no easy task. It's not uncommon for me to spend 40+ hours writing and preparing a single 30 minute talk :(

I'm hoping that over the next year, I'll have sufficient practice to make this process a bit more automatic and easier :)

You Can Do a Lot in a Year (or Two!)



In the past year, Stormpath has grown an incredible amount.

Since I've joined, the company has launched several huge features, completely redone our entire product UI, added tons of new users, picked up lots of revenue, and also expanded our footprint into many new developer communities.

My own, proudest achievements to date, include:

- Building an incredibly tightly integrated Flask-Stormpath (https://github.com/stormpath/stormpath-flask) library which makes building secure Flask web apps drop-dead simple. This library generates web registration and login pages, handles social login via Google and Facbeook, and provides really convenient methods for accessing and storing user data. It does a lot of things that no other Flask tools out there do, and has one of the nicest, cleanest APIs I've ever written.

- Helping to revamp Stormpath's Python (https://github.com/stormpath/stormpath-sdk-python) and Django (https://github.com/stormpath/stormpath-django) libraries to improve the quality, look, and feel of the library for developers. It's now not only a useful library, but an elegant and simple one =)

- Building out a full featured Express-Stormpath (https://github.com/stormpath/express-stormpath) library which makes building secure Express.js web apps simple. Just like with our Flask integration, our Express.js integration does incredibly powerful things that no other apps out there do, and honestly saves developers using it a ton of time and energy.

- Pair programming with nearly 50 separate users, helping them solve real world problems related to user authentication. Each time I do it I have a ton of fun, and get to see how REAL APPS ARE GETTING WRITTEN out there in the *real world* by real people. It's exciting!

- Building out an analytics pipeline along with our core engineering team, which generates all sorts of cool company statistics, and makes tracking progress possible.
- Building out brand new developer library and product documentation, and developing new systems to automate improve the documentation process.

Key Takeaways

If you're just getting started with an evangelism role, before diving into your job, take a few moments to evaluate where your product / service is at in a realistic way. What's the most valuable thing you can start on immediately?

One question you should always be asking yourself as you work on projects is: *"How will this help a developer trying to do X?"* If the answer is *"it won't"*, you might want to take a step back and reconsider.

Track your progress (*loosely*) via company metrics:

- How many new developers make API requests to your service?
- How many of these developers stick around after the first month or so?
- How many new signups do you get per month?
- What programming language is most popular amongst your userbase?
- How is your API usage distributed? Do you have a few customers generating all your requests? Or is it more evenly distributed?

If you do things right, you should hopefully see:

- An increasing amount of user growth every month.

- An increasing amount of API usage in each programming language when you improve your developer libraries.
- An increase in API activation rates (*how many new signups actually try out your API service?*) when you simplify on-boarding stuff.

And – I think that's just about it.

I plan to dive into more in-depth articles about evangelism in the future, to share what things specifically have worked well (*and not so well*) for me at Stormpath. So, if you're interested, be sure to follow along =)

PS: If you read this far, you might want to follow me on twitter (https://twitter.com/rdegges) or github (https://github.com/rdegges) and subscribe via RSS (/feed.xml) or email below (*I'll email you new articles when I publish them*).