The Hat Man

Author: Andrew Kannan

Claps: 318

Date: Feb 2, 2019

One clever little piece of software weâ€™ve built here at Klaviyo is an open-source slack bot we call â€œThe Hat Man.â€� This is a story about The Hat Manâ€™s inception.

When I joined Klaviyo in February of 2017, there were eight engineers total â€" a far cry from the 30+ engineers on our engineering team now. We had just started using AWS auto-scaling groups for many of our workloads, and started running more servers as we grew. In order to keep code in sync across all of our boxes, we deployed via a Fabric script, which would ssh into each box, pull down master from the git repository, and restart all of our worker processes. Typically weâ€™d run this script in parallel, so we could hit many boxes efficiently. With such a small team, this was sufficient.

Hat On

But being a fast-growing startup makes this type of deployment challenging. Teams work at a fast pace, and multiple deployments in a day are not out of the norm. As we added more engineers, deploy collisions became increasingly likely. While we could recover from these deploy collisions, they often left our workers in a bad state, and required a good amount of time to recover from. After a few instances of these deploy collisions, and with Black Friday 2017 approaching, we knew we needed a better solution.

There was a slight problem â€" we didnâ€™t have the time prior to Black Friday to completely overhaul our deploy process. While building out a CI pipeline would have been the most optimal solution (and one in which weâ€™ve made marked progress), our engineers at that time were focused on the problems that arise with increased scale, and handling the load that we expected on Black Friday. The key issue at hand was concurrent deploys, so we decided to institute a â€œone deploy at a timeâ€� policy.

Hat Off

To facilitate this process we had many ideas. The easiest solution we came up with was the notion of a â€œdeploy hatâ€�. Any engineer that wanted to deploy had to wear the deploy hat. Since there was only one deploy hat, we were guaranteed that concurrent deploys could never happen. Enter **Hat Man**.

The Hat Man started as a thin layer around the Slack python SDK. There was essentially just a script that set up the `slackclient` package with the proper credentials and used `slackclient.rtm_connect()` and `slackclient.rtm_read()` in a loop to read all messages in every channel the bot was present. If the message started with â€œ@the-hat-man,â€� the Hat Man would interpret it as a command. Two commands were available â€” on and off â€” for putting the hat on and taking the hat off. Slack provides two APIs â€” the Events API, where you can subscribe to specific events and they are pushed to you via HTTP, and the Real-Time Messaging (RTM) API, where you can open a Websocket and poll for a superset of the information available in the Events API. As we did not want to open a public-facing HTTP endpoint for this bot, we chose to use the RTM API â€” again, this was supposed to be lightweight.

In order to store state about who had the hat, we used the small python ORM `peewee`. We didnâ€™t want something as heavy as SQLAlchemy or Django to simply store a tiny bit of state, so we relied on `peewee`. We defined models for peewee and kept them separate from the simple slack-read busy loop. Our first model looked something like this:

When a user tried to run â€œonâ€�, weâ€™d check the database to see if anyone had the hat, and if not, created an entry saying that user had the hat. When that user ran the â€œoffâ€� command, weâ€™d set an end timestamp on the row. If a row didnâ€™t have an end timestamp, it meant that the hat was still out there, and could not be assigned again.

Hat Pool

The Hat Man continued like this for weeks, until Black Friday finally arrived. Our engineering team arrived early in case of an emergency, but the day went quite calmly. This was when The Hat Man took on a whole new life.

While we definitely had other work to do, we didnâ€™t want to start on any projects that Black Friday could potentially interrupt. The Hat Man was a perfect candidate for a refactor with minimal impact on production systems. We decided to make The Hat Man an extensible python slack bot. Now, I know a lot of extensible python slack bots already exist. If you go to this link: https://api.slack.com/community#python you can see just how many exist. But hacking away at little mini projects is fun, and we wanted to build upon the little slack bot we already started.

Hat Queue

Our stack remained the same â€” we still used python with the Slack SDK using the RTM API, and peewee. We just made our code a little bit more intelligent. Instead of hard coding two commands, we defined a base command class which subcommands could inherit from. Each function defined in the subcommand would become a user-facing hat man command. The â€œhatâ€� related commands moved to their own â€œhatâ€� module. Our models were also greatly expanded to support newly added â€œhatâ€� functionality, including a queue, a â€œpoolâ€� (so multiple people could share the hat), and a â€œforce offâ€� command (for when people forgot to take off the hat).

Hat Force Off

If youâ€™re interested in seeing The Hat Man code, itâ€™s open-source and available here: https://github.com/klaviyolabs/TheHatMan. Itâ€™s not built with nearly the same quality as our production systems, but it was a fun side project where our developers can add new easter eggs. Feel free to take The Hat Man code and use it for yourself, but weâ€™d also strongly suggest considering some of the other more robust alternatives.

While heâ€™s still in use today, The Hat Man will likely be deprecated this year. Weâ€™ve been improving our build systems to avoid some of the problems that The Hat Man was created to address. Weâ€™ve replaced our fabric script with a new deploy infrastructure thatâ€™s way more scalable that also prevents concurrent deploys. And as we continue to break our monolith into microservices, more and more of our stack will be under CI with Jenkins. But the Hat Man proves that a simple hour long project can pay off in spades in developer velocity and reliability. And even when heâ€™s gone, The Hat Man will always have his little spot in Klaviyo history.