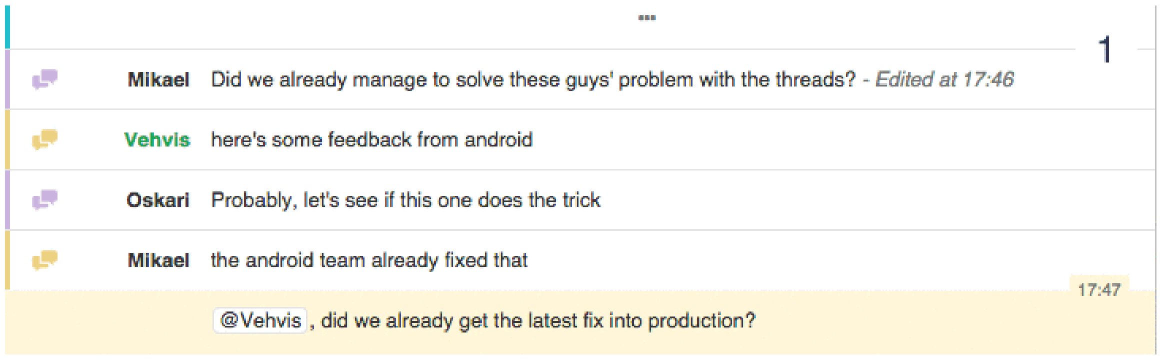
# CHATOPS: EVERYTHING ABOUT DEPLOYMENTS RIGHT INSIDE YOUR CHAT

Nov 11, 2014 — by Otto Vehviläinen

Flowdock is a sophisticated application made up of dozens of individual services. Keeping it up and running with no downtime while practicing continuous delivery is no easy task. We recently identified some pain points in our deployment process – such as limited visibility into the current state of our application – so we set out to improve it by turning to ChatOps.

Our goal was to be able to do something like this:



Mikael Did we already manage to solve these guys' problem with the threads? - *Edited at 17:46*

Vehvis here's some feedback from android

Oskari Probably, let's see if this one does the trick

Mikael the android team already fixed that

17:47

@Vehvis , did we already get the latest fix into production?

Below is a summary of what we learned, our solution and results.

## Improving deployments with ChatOps

GitHub has been pioneering ChatOps – a way to automate most ops-related tasks with a chat bot. They've also done some awesome work building components related to the workflow, Hubot being the most popular example. We've had Hubot running in our own flows for a long time, but he's mostly been idle, waiting for something meaningful to do.

We decided to build our new deployment process based on Hubot and ChatOps. This is what we had in mind:

- All deployments are initiated in Flowdock by chatting with Hubot. This provides visibility into who is deploying what, in real-time. Additionally, the chat transcript functions as an audit log.
- Hubot creates deployments using the GitHub deployments API. This leads to a single, easy to access source for all information about deployments.
- Heaven receives webhook events about initiated deployments and performs the actual deployments using capistrano. Only the deployments server needs SSH access to our production application servers.
- The status of deployments are sent back to Flowdock, making them visible and searchable for everyone.

Additional features included:

- Automatic deployments to our QA environment when a master branch is updated and our builds succeed and tests pass.
- Automatic checks that all environments are up to date with master.

## The workflow, component by component

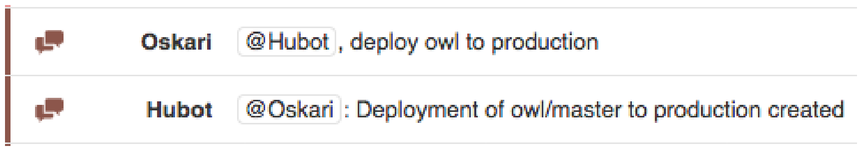
The entire process has quite a few moving parts:

## First we need a Hubot to chat with

The first essential part is Hubot, but luckily it's pretty easy to integrate it with Flowdock.

## Orchestrating everything with hubot-deploy

Hubot-deploy is the first deployment-related part of the process. It defines the available deployment targets and provides the necessary Hubot commands to initiate deployments. The script is configured with a simple apps.json file that tells Hubot about the deployable applications, their repositories and how they should be deployed. When it's up and running, just start deploying:



**Oskari** @Hubot, deploy owl to production

**Hubot** @Oskari: Deployment of owl/master to production created

This creates a deployment in the corresponding repository. As an added bonus, it will check for a green commit status (build is successful) and will even automatically merge the master branch into the branch being deployed if the branch is lagging behind. Additional features include the ability to specify a different deployment command (`deploy:migrations`), forcing a deployment even if the build is failing (`deploy!`), deploying a branch (`app/branch`) or specifying the target environment or servers (`to production/frontend`).

## Deployments API to glue everything together

GitHub's deployments API is still in beta, but it can already do many cool things. The API acts as glue that holds the process together, but it has other benefits as well: having all of our deployments recorded in the GitHub API makes them easily accessible for other components, and allows us to gather better status information across our infrastructure.

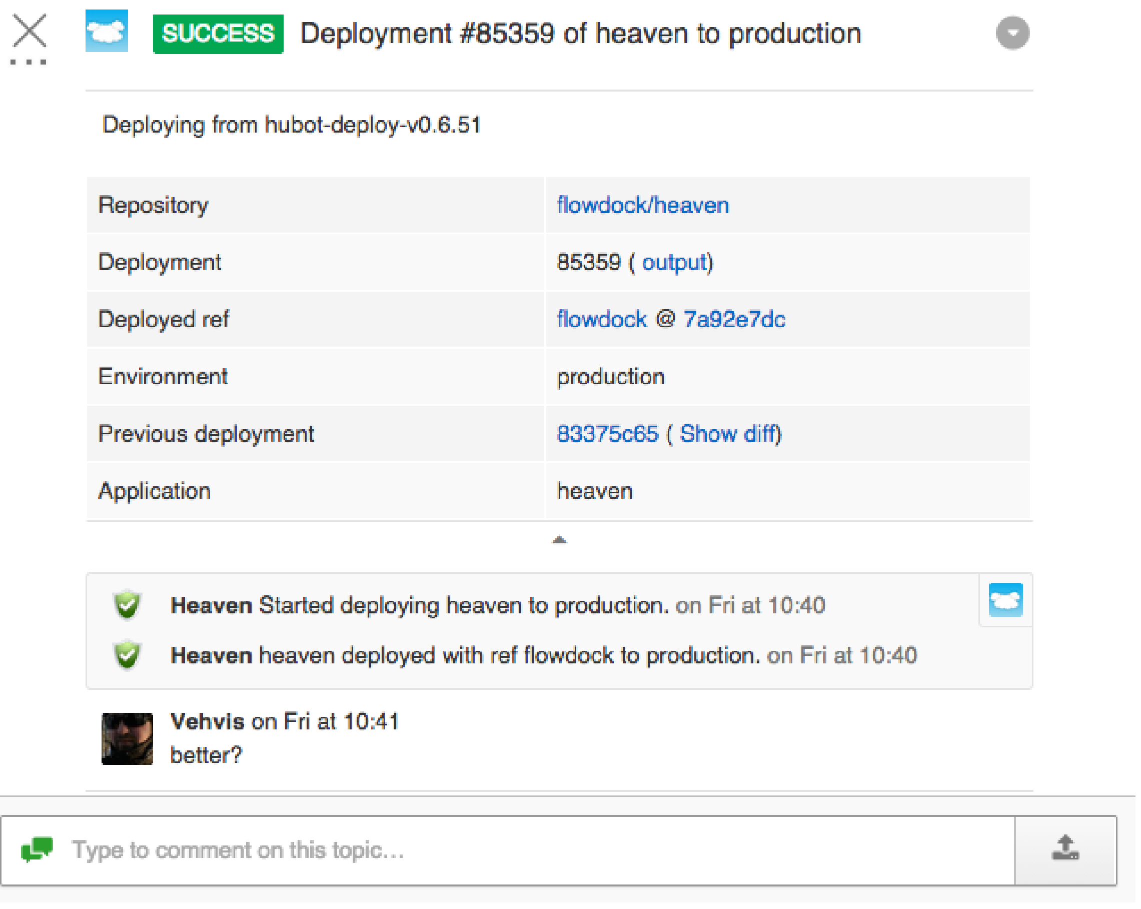
A webhook in GitHub posts to our deployment server whenever a deployment is created. The deployment server then takes over the process and updates the state of the deployment in GitHub as it progresses.

## Heaven for performing the deployments

Heaven is the deployment server that performs the actual deployments. It listens for GitHub deployments and deployment status webhooks, and executes the deployment tasks as background jobs. Heaven supports multiple deployment providers, meaning you can use it to deploy to Heroku or by using capistrano or Fabric, to name a few options.

Since Heaven is designed to be run in Heroku, the capistrano provider was a bit limited. We created a capistrano provider that supports both cap2 and cap3 through the use of bundler. It's now included in Heaven. Once we had Heaven up and running, we needed to strip out most of our custom capistrano configurations since Flowdock notifications and requirements for deployments were now handled elsewhere. Our capistrano recipes ended up being significantly leaner.

We also reworked the Flowdock notifier to take advantage of our most advanced features. Heaven now supports our new, enhanced (beta) Threads API. All the activities and discussions related to a single deployment now appear in one thread, with the current status of the deployment updated in real-time at the top of the thread.


SUCCESS Deployment #85359 of heaven to production

Deploying from hubot-deploy-v0.6.51

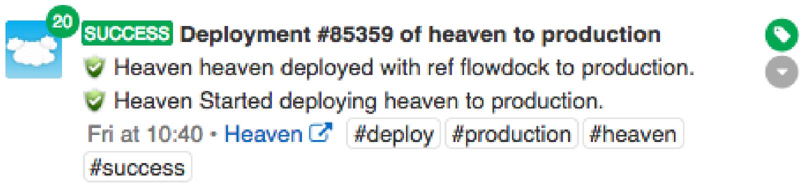
| | |
|---|---|
| Repository | flowdock/heaven |
| Deployment | 85359 ( output) |
| Deployed ref | flowdock @ 7a92e7dc |
| Environment | production |
| Previous deployment | 83375c65 ( Show diff) |
| Application | heaven |

**Heaven** Started deploying heaven to production. on Fri at 10:40

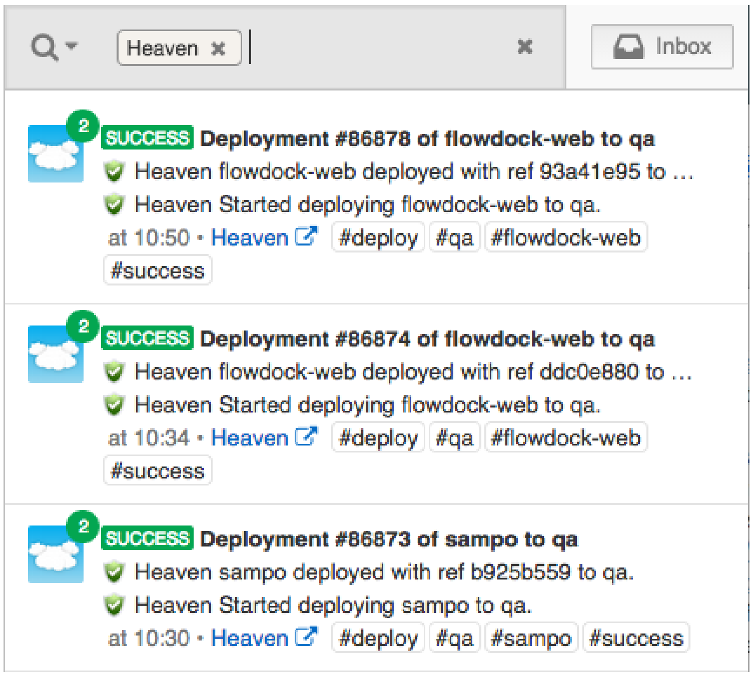
**Heaven** heaven deployed with ref flowdock to production. on Fri at 10:40

**Vehvis** on Fri at 10:41
better?

The deployment updates are displayed in the team inbox, along with the latest state of the deployment. Deployment updates no longer clutter up your chat:


SUCCESS Deployment #85359 of heaven to production
Heaven heaven deployed with ref flowdock to production.
Heaven Started deploying heaven to production.
Fri at 10:40 • Heaven #deploy #production #heaven #success

Of course, all your deployments are available via search:



SUCCESS Deployment #86878 of flowdock-web to qa
Heaven flowdock-web deployed with ref 93a41e95 to …
Heaven Started deploying flowdock-web to qa.
at 10:50 • Heaven #deploy #qa #flowdock-web #success

SUCCESS Deployment #86874 of flowdock-web to qa
Heaven flowdock-web deployed with ref ddc0e880 to …
Heaven Started deploying flowdock-web to qa.
at 10:34 • Heaven #deploy #qa #flowdock-web #success

SUCCESS Deployment #86873 of sampo to qa
Heaven sampo deployed with ref b925b559 to qa.
Heaven Started deploying sampo to qa.
at 10:30 • Heaven #deploy #qa #sampo #success

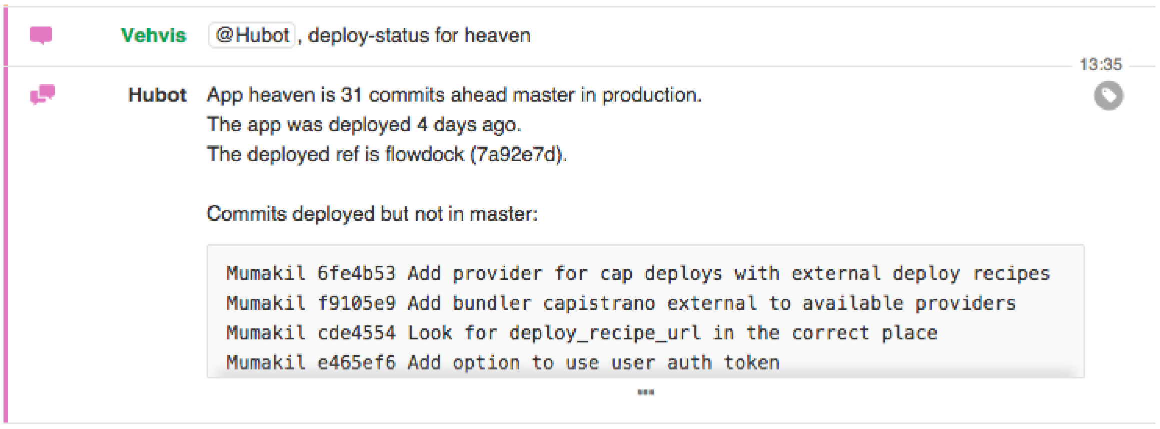
This comes in handy when you're debugging whether a code change caused a production issue.

## Deploy automatically with Hubot-auto-deploy

Hubot-auto-deploy provides the necessary commands to configure automatic deployments. All of our applications are now automatically deployed to our QA environment by the GitHub autodeploy service when the master branch is updated and has a green commit status. Using Hubot-auto-deploy made configuring the automatic deployments really simple, and also lets us disable them when we want to run other branches in QA for prolonged periods of time.

## Hubot-deploy-status for querying rich deployment status information

Hubot-deploy-status is a Hubot script that lets us see how our deployed builds differ from the repository's master branch. We use it to communicate what has not yet been deployed:


Vehvis @Hubot , deploy-status for heaven

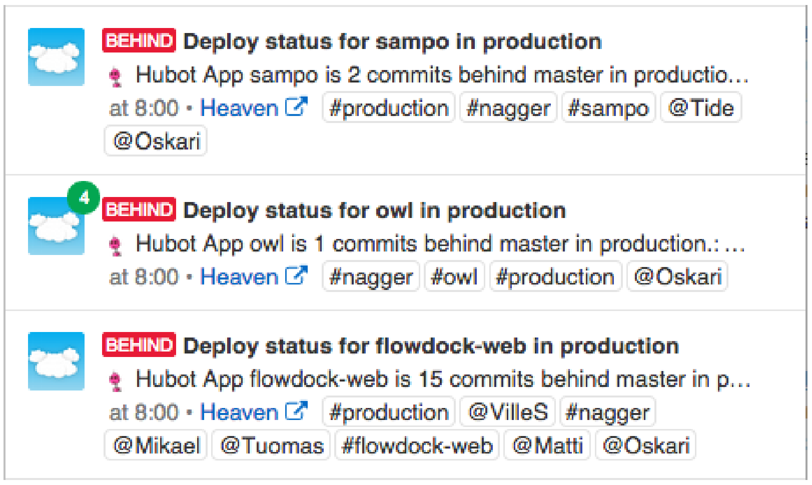
13:35

Hubot App heaven is 31 commits ahead master in production.
The app was deployed 4 days ago.
The deployed ref is flowdock (7a92e7d).

Commits deployed but not in master:

```
Mumakil 6fe4b53 Add provider for cap deploys with external deploy recipes
Mumakil f9105e9 Add bundler capistrano external to available providers
Mumakil cde4554 Look for deploy_recipe_url in the correct place
Mumakil e465ef6 Add option to use user auth token
```

The script also runs automatic checks from time to time. Our production environment is checked every morning for pending commits and the results are posted to our flow's inbox:


BEHIND Deploy status for sampo in production
Hubot App sampo is 2 commits behind master in productio…
at 8:00 • Heaven #production #nagger #sampo @Tide @Oskari

BEHIND Deploy status for owl in production
Hubot App owl is 1 commits behind master in production.: …
at 8:00 • Heaven #nagger #owl #production @Oskari

BEHIND Deploy status for flowdock-web in production
Hubot App flowdock-web is 15 commits behind master in p…
at 8:00 • Heaven #production @VilleS #nagger @Mikael @Tuomas #flowdock-web @Matti @Oskari

We want to encourage our team members to deploy code often and in small batches. There are three main reasons for this: improvements should be pushed out to users as soon as possible, rollbacks are less risky, and tracking down bugs and performance issues is easier if you don't deploy ten different features at once. If Hubot knows your GitHub username, it will tag you in the message when some of your commits are pending, alerting you with a Flowdock notification.

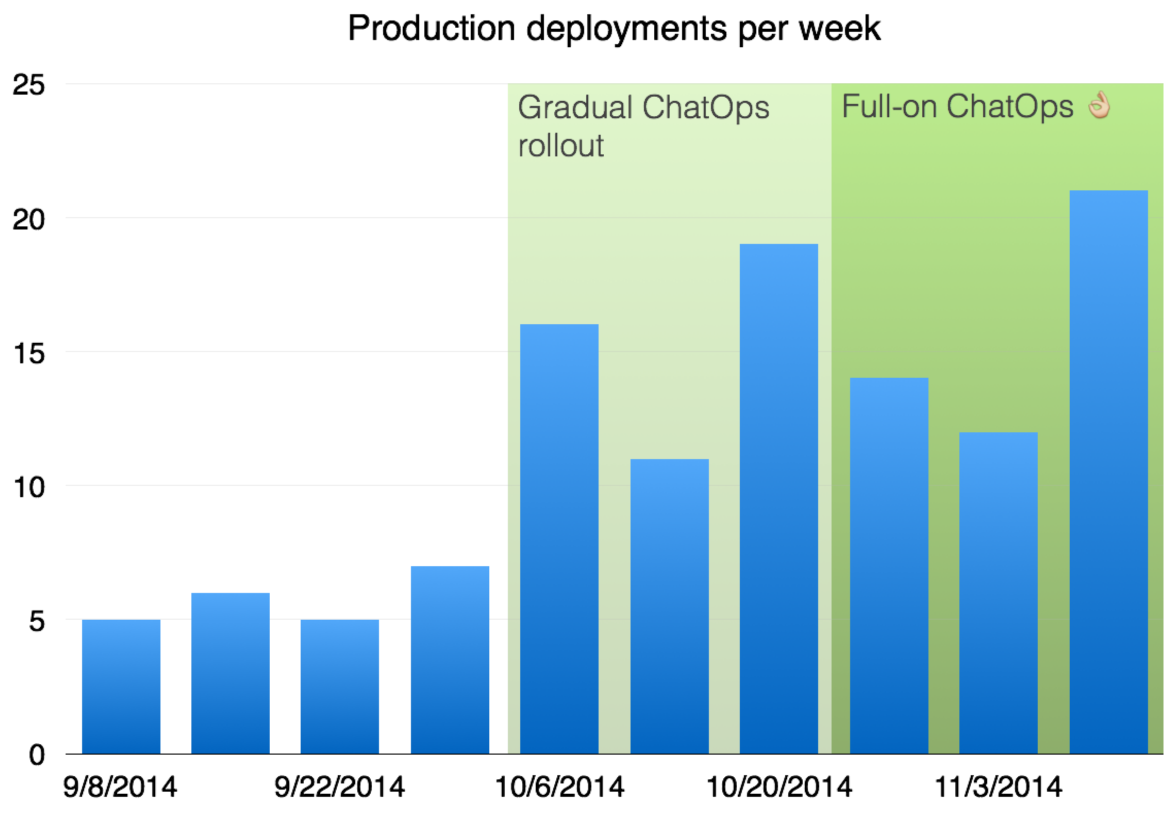
## Expanding Hubot usage

Now that we're using Hubot more and more as an important part of our workflow, we have started expanding its capabilities. It's now possible to query the status of our applications in different environments (service masters / DB states…) using our hubot-command script and to fetch graphs from Graphite, right from within our development flow.

As part of our effort to build all of this, we also gave our API packages and other components quite a bit of love. Be sure to check them out if you want to build custom Flowdock integrations.

## Future development

We've only been using this workflow for a short time, but it has already greatly helped our continuous delivery process. As can be seen, deployments are up by 100-200%:



However, there's still lots to do. Our new API is already aware of external data (context), so we could do things like:

- Just say "ok deploy" to the deployment nagger message, and Hubot automatically starts the pending deployment.
- Retry failed deployments by just commenting "retry" in a deployment thread.

Unfortunately, both of these would require forking lots of code in Hubot and related scripts since most of the chat networks that Hubot supports don't provide this rich context.

## The end result?

We're big fans of continuous delivery. We deploy code to production multiple times a day, but before these new tools, we felt our deployment process had some shortcomings:

- Even though we received notifications about finished deployments in Flowdock, we didn't get visibility into when deployments were started (and by whom). Now we instantly see when deployments start and who started them, leading to deployment-related discussions in context. Deployment information is no longer siloed away in command-lines and terminals.
- When using capistrano to deploy, only those who had SSH access to our production environment could deploy. We've now been able to decouple being able to deploy from production access, which is great for security.
- We had a few issues when nobody noticed that a component was lagging behind master in production by a few days. With automatic deployments to our QA environment and automatic daily status updates for each component, we can make sure everything's running with the newest code.
- Information about deployments was hidden in our application servers, making it hard to access. With several dozen individual services that make up the full Flowdock application, coordinating the whole was starting to become difficult. With the deployment statuses in GitHub, we can build better tooling and have real-time visibility into what's currently running.

All in all, the improvements made it easier for us to manage the entire stack and removed a lot of manual work. An even better benefit is the reduced mental overhead thanks to being able to trust that every service is up to date. Deploying is also very easy: even our UX dude feels comfortable deploying code to production without the stress that comes with typing (possibly disastrous) commands into a production console.

Are you interested in taking this setup to the next level? We happen to be looking for a DevOps-minded person to join our team in Helsinki!

## Big thanks

All of this wouldn’t have been possible without atmos, who has developed most of the components that we use, and the entire GitHub deployments API team. A big “Thank You!” to all of you is in order!

ABOUT THE AUTHOR

Otto Vehviläinen
@Mumakil, vehvis@flowdock.com

*I'm a developer at Flowdock. I work with both backend and frontend, so it's all about Rails, Node.js, CoffeeScript and Backbone for me.*

3 Comments The Flowdock Blog 1 Login

Recommend 7 Tweet Share Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

**OlivierJ** • 4 years ago

I can't believe this post has no comments :) Very cool use of ChatOps. Big fan here as well, it has changed the way our team operates.

2 • Reply • Share ›

**c0de** • 2 years ago

You should check out https://github.com/adnanh/w...

• Reply • Share ›

**Luis De Siqueira** • 4 years ago

Huge fan

• Reply • Share ›

Subscribe Add Disqus to your siteAdd DisqusAdd

## Leave a Reply

Email address will be published with comment.

Write a comment

Name

Email address

POST COMMENT

Flowdock.com Features Pricing About API Twitter Facebook