Building a User-Centric Product Team
• The platform for application developers.

• Offers a variety of developer tools and services.

• We’ll be using the Heroku Postgres team as our case study.
Heroku Postgres
The SQL database service for developers.

Sign up for free

Operational Expertise, Built In

Whether it’s an index that’s not being used, security patches that have to be applied, or guidance relevant to ensuring your database is performing well, we’re here to guide you along the way. Our guidance is the result of running the
500,000 - 1,000,000
Postgres databases
ops, engineering, product, advocacy & support...

(not handling: front-line support, marketing, or sales)
10 people
~100,000 databases per person
User-Centric Team?
“Sure, we can do that.”

– Every Founder to their First Customer
All the major and many of the minor living branches of life are shown on this diagram, but only a few of those that have gone extinct are shown. Example: Dinosaur.
Organization Functions

- Product Engineering
- Sales
- Business Operations
- Marketing
- Customer Success
- People Ops
- Technical Operations
- Business Development
- Office Management
Organization Functions

- Product Engineering
- Sales
- Business Operations
- Marketing
- Customer Success
- People Ops
- Technical Operations
- Business Development
- Office Management
Focus

- such a huge topic
- focus on scaling engineering/customer interactions
- particularly support and operations
- how we adapted our culture and practices
How We Think About It
Know the user.
Scale is automation.
Design for failure.
Make the right thing the easy thing.
Not everything went entirely according to plan...
Go Looking for Trouble
“Houston, we’ve had a problem…”

– Jack Swigert (Apollo 13)
The most dangerous problems often go unrecognized.
The Case of the Disappearing Service
“Shouldn’t those new servers be online by now?”
The Problem

- Servers booting would rely on an arbitrary set of services in their userdata.
- Sometimes these services would go offline or change how they worked.
- Inability to boot new capacity would result.
Our Solution

• Design for failure.
• Don’t require external services to enter service.
• Build ready-to-go machine images (AMIs).
• Keep images simple so that they change rarely.
• Manage servers “outside-in”.

The Case of the Boiling Frog
“So, how was on-call last week?”
The Problem

- The on-call experience would get steadily worse any time it wasn’t our focus.
- On-call engineers could no longer fix most issues that bothered them. (Code-base too big.)
- Engineers would simply triage bugs and endure pain during their rotation.
Our Solution

- *Scale is automation.*
- Establish pager burden goals as top-level team priority.
- Build a dashboard which tracked all alerts over time.
- Review the dashboard weekly and prioritize work every cycle.
(the dashboard)
My Advice?

• Review pager burden and team-wide weekly.

• Consider mercy for people who are more bothered by being on-call. (I never really minded it.)

• You’ll need an automation framework.
The Case of the Broken Feature
“I went renegade and wrote a fix for [your broken feature].”

– Support Team Member
[six months later...]
Support was responsive & satisfaction was high.
...but the product experience was awful!
Being great at answering tickets is just losing with style.
The Problem

• Support was shielding engineering from the defects in order to do their jobs.

• Engineering was rotating through support detail.

• Nobody anywhere understood the magnitude of the problem.
Our Solution

• *Know the user.*

• Embed one support person with the engineering team. Make them an expert on the product domain.

• Each week, analyze and report on support issues and trends.
An early support report

By The Numbers:

<table>
<thead>
<tr>
<th>Type</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoD tickets</td>
<td>62 / 341 (18%, down from 26%)</td>
</tr>
<tr>
<td>Ticket sample size</td>
<td>20 (30%)</td>
</tr>
</tbody>
</table>

Survey

<table>
<thead>
<tr>
<th>Source</th>
<th>Count</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>100 - 500</td>
<td>6</td>
<td>30%</td>
</tr>
<tr>
<td>500+</td>
<td>14</td>
<td>70%</td>
</tr>
</tbody>
</table>
My Advice

• Track defect remediation (not just C-SAT).

• Make sure support has a strong, human relationship with engineering.

• Maintain slack in your team to respond quickly to minor or emerging issues.
The Case of the Squeaky Wheel
“You know, people keep asking me why their queries are so slow.”
“UX Issues”
aka “Pilot Error”
The Problem

• Users didn’t really know much about their database.

• Growth caused problems because they weren’t familiar with things like indexes or locks.

• We spent an increasing amount of time doing basic DBA work.
Our Solution

• *Make the right thing the easy thing.*

• We built a variety of customer-facing tools like pg-extras, pg:diagnose, and the slow queries analysis tool.

• A self-rescued customer is one that never has to open a ticket in the first place!
(The easiest thing.)
The Case of the Mysterious Software
“That? Oh, the contractors wrote that.”
The Problem

- Outside people aren’t subject to your culture.
- Projects run externally don’t follow your practices.
My Advice?

- Great for short-term work (like a marketing site.)
- Great for OSS community work (like Postgres.)
- Also great as a way to lure in candidates.
- Don’t let uninvested people own key code-bases.
Best Practices
My Favorite

Best Practices
Continuous Deployment

- Shipping the service must be trivial.
- Ship very small patches very often.
- Avoid long branches.
- Protect new work behind feature flags.
The Sweeper

- One engineer at a time
- No assigned work
- Carries the pager
- Helps with tickets
- Buffers interruption & hands off as appropriate
The Sweeper

• One week rotation.

• If it gets rough, someone steps in to give them a night off.

• PagerDuty for alerts.

• 1-2m “rest time” in chat before the real page. (We are a global team.)
Burn Party

- Celebrates retired infrastructure / code.
- Build a fire.
- Bring an effigy.
- Say a few words and torch it.
- (The thing must be totally gone to qualify.)
The Toolbox

• Heroku for running code. (It’s really great.)

• Git with short-lived branches, deployed constantly.

• GitHub for (100s of) source repos, bug tracking, and code review.

• Trello for project tracking.

• Twice-annual off-sites for long-term / big-picture thinking.

• PagerDuty for getting woken up.
In Summary
Go looking for trouble.
Get close with your users.
Users don’t make mistakes, we give them bad software.
@pvh / pvh@heroku.com

fin

(If this sounds fun, we’re hiring!)