How ReadMe Went From SaaS To On-Premises In Less Than One Week

readme

AT A GLANCE

Category:
Content Management

Customer Since:
2015

Solution:
Replicated Core

Use Case:
Healthcare

Best Features:
- Updates
- Installation

Results:
- 10x deal
- Easy to deploy
- Happy enterprise customer

The Story

Over 5,000 companies use ReadMe as their content management solution for developer facing resources like documentation and sandboxes One feature request ReadMe kept getting was for it to be available to be deployed as a single-tenant, private instance within a customer's own servers, aka "on-premise." It made sense as a feature request (HIPAA compliance, secret projects, etc), however it was something they felt they wouldn't be able to do until they had a whole Enterprise team. So, they told anyone that asked for it that they unfortunately weren't able to do it. They got enough requests, however, that we decided to investigate what it would take to accomplish. It turned out to be much easier than they expected.

Dockerizing ReadMe

They knew the easiest way to deploy ReadMe would be to Dockerize ReadMe. Their original plan was to just send out Docker containers that people could install locally, so they started there. Since they weren't currently using Docker, they had to containerize their app. It sounds a lot scarier than it really is. They created a Dockerfile that looks like this:

```
FROM node:wheezy
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
COPY package.json /usr/src/app
COPY . /usr/src/app
RUN npm install -g gulp
RUN gulp deploy
ENV NODE_ENV "enterprise"
ENV MAILGUN_USER "postmaster@readme.io"
ENV MAILGUN_PASS "********"
EXPOSE 3000
CMD ["node", "server.js"]
```

It's basically just a deploy script, with some environment variables and configuration mixed in. Overall, Dockerizing their app took them about 2-3 hours. The biggest blocker was databases. They previously used Compose.io for their backend database, so getting Mongo running inside Docker was the hardest thing.

> On-premise was a simple feature to add that directly impacted our growth. It's had the single biggest ROI for us of anything we've done.

Gregory Koberger
Founder, ReadMe

Managing Dockerfiles

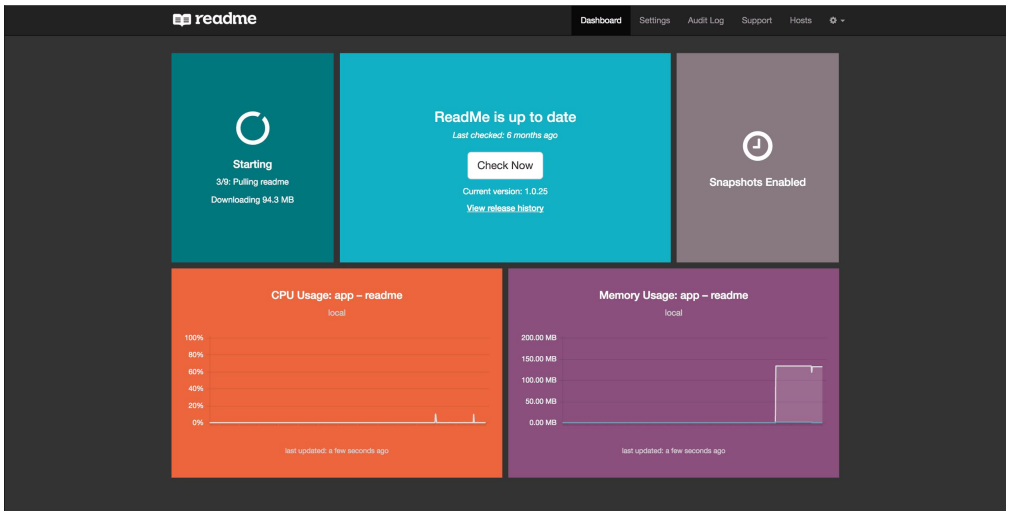
They originally thought about doing it themselves by sending customers a Docker container. However, the scope quickly spiraled: licensing, configuration, orchestration, updates, backups, reporting, monitoring, debugging…. there was a lot to worry about. Also, their first customer was pretty adamant about ensuring the integrity of their private environment, meaning they wouldn't have access to the machine it would be running on. Without being about to view the logs or the server, they felt like they'd be too in the dark.

They found Replicated and noticed that a bunch of companies they liked (NPM, Travis, etc) were already using the platform. They decided to give it a shot and see how fast they could get up and running. Greg Koberger describes the process by saying, "It's great; you just provide Replicated with a Dockerized version of your app and they provide the features that make it an enterprise-ready, installable application. After spending a few hours getting our services Dockerized, it took us less than a day to integrate and get the on-premise version of our product ready to go."

Deployments

For their enterprise customers, the experience is pretty simple. The enterprise IT admin runs an install command on whatever server they want to deploy to, and system sets itself up. Their customers get it set up and running almost as quickly as signing up on the SaaS site.

They didn't want their customers to be stuck on old builds of ReadMe, which would happen if they just sent out a plain Dockerized version. Replicated made it easy for Readme to set policy for each individual enterprise customer's license. For smaller customers they can set policy to auto-deployed any updates delivered by ReadMe, or for bigger customers they can provide the ability for each update to be manually approved. Either way, upgrading is incredibly simple: at most it's one-click for the customer. Their SLA dictates that they don't support older versions (except for security updates), and they haven't had any issues with this yet. Most of their customers are happy to stay up-to-date; they just want to be in control of pressing the button.



Customer facing enterprise admin console.

Managing Environments

They use the same codebase for their on-premise deployments. Much like how they have a production, staging and development environment, they also have an enterprise environment. There's two major differences in their enterprise builds:

- **Customer-entered environment variables** - These are things specific to each deployment, such as the URL or their Mailgun credentials. Replicated lets the customer manage these variables, and they just read them the same way as normal environment variables.
- **Turning on/off irrelevant features** - Not every feature, such as pricing, is relevant for enterprise builds. Enterprise builds also get some additional features that normally only their support staff would see, such as the ability to toggle on beta features or view boring metadata they normally hide away. They also didn't want it to phone home at all, so those features were disabled.

Third Party Apps

The ReadMe team are big fans of using third-party services for as much as possible: Stripe for billing, Mailgun for emails, Segment for analytics, etc. (full stack listed here). This is great for the public cloud version of your app, but becomes problematic when going on-premise (as your end customer will likely not want to set up a litany of 3rd party services to run your app). So, they split things into two categories: things that could be removed (analytics, billing), and things that could be replaced (emails).

For the former, they just put those features behind a flag and hid them when the environment was Enterprise. They were already using basic feature flags to hide things like analytics – they're just "if" statements in their code for the most part. For the latter, they allowed the customer to configure the application with their own keys (you can also abstract things like SMTP out to allow your customer to use any SMTP server if needed).

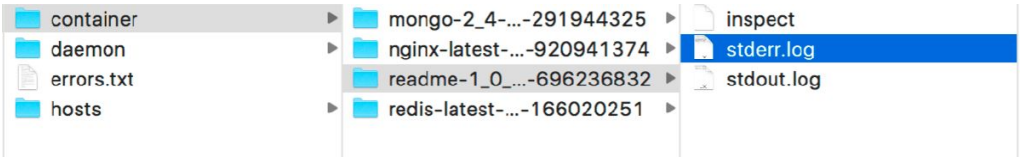
Communication and Support

One thing they failed at early on was encouraging their customers to come to them as soon as there was a problem. Since they couldn't monitor the servers the way they do in production, they had some issues early on with things breaking and them not knowing about it. It caused a bit of frustration on the customer's end and they were blissfully unaware (for a time).

So, they've worked to be better at communicating with customers using their on-premise version. Replicated lets you do some basic monitoring (since anything more would defeat the purpose of being on-premise), and we've gotten better at spotting issues.

Companies get a nice dashboard, so they know what's running and what's broken. They have four services running (main site, NGINX, Mongo and Redis), and they're easily monitored:

When you need to debug, you can have the company send you a “support bundle” zip file that contains logs for all these running services:



On their end, they budgeted two weeks of a developer’s time for each install. It’s a lot of time, however once it’s all up and running it tends to work well in the future. If they end up doing more of these, I’d likely hire an engineer to do this full-time (and potentially on-site). They haven’t had a need for that yet, though. Greg recommends doing your first installation with a friendly company. That way you can polish any rough edges in the process or product, before installing it at larger companies. Like anything, the first time never goes as smoothly as you expected!

> If you have customers asking you for on-prem, you don’t have an excuse... it’s a lot easier than you’d think!

Gregory Koberger
Founder, ReadMe

Enterprise Contracts

They added a line to their ToS that everyone had to agree to before they could install the Dockerfile.

> *You must not attempt to reverse engineer, read, or modify any ReadMe code, or any code in a third party package being used by ReadMe. This includes any code, configuration, or tools specifically related to the build process (eg. Dockerfile).*

They don't obfuscate their code, and trust their customers to not violate the ToS. In the future, if this becomes a problem, they'll start minifying their Node before shipping. They also created a support SLA for their companies using their on-prem version. They give everyone their direct phone number for 24/7 support (luckily this hasn’t been abused!), and guarantee 24-hour responses for non-vital issues. It’s important to be proactive: since you can’t see the site or detect issues, early on it’s important to check in every few days to make sure there’s no issues.

Why You Should Too

Companies often chase after that elusive next feature which will change everything and make revenue shoot up. And that feature doesn’t exist… with one exception! On-premise was a simple feature to add that directly impacted their growth. They went from their normal $59 plan to being able to sell their product for two orders of magnitude more. It’s had the single biggest ROI for them of anything we’ve done.

As Greg says, “If you have customers asking you for on-premise, you don’t have an excuse… it’s a lot easier than you’d think!”

To get started, just sign up at www.replicated.com or email contact@replicated.com