portworx lix

Case Study: Lix Solves Cassandra, Postgres, and Elasticsearch Ops on Kubernetes to Build World's Best Study Experience

CHALLENGES

- Lix customers demand a fast, responsive platform to help them succeed with their studies
- Containers promised to offer an easier way to build a high quality platform, but only if it could provide a solution for running and managing data services like Elasticsearch, Cassandra, and Postgres

SOLUTION

- Leverage Kubernetes to provide an environment that easily scales with the growth of the Lix platform
- Portworx Enterprise for cloud native storage and data management

RESULTS

- Portworx saved Lix time by allowing it to focus on creating the world's best study platform and not managing storage
- Portworx enabled Lix to achieve automation via containers without sacrificing performance
- Portworx increased Lix's operational efficiency by automatically handling common failures that occur in a dynamic server environment like server failures and network partitions, enabling them to focus on their customers

KEY TECHNOLOGIES DISCUSSED

Infrastructure – bare metal hosting on Packet
Container Runtime – Docker
Orchestration – Kubernetes
Stateful Services – Cassandra, Postgres, Elasticsearch

Lix is a study platform revolutionizing the way college students access and use textbooks. We sat down with Simon Stender Boise to learn about how he overcame challenges on Kubernetes with slow failover, label management, and stateful recovery. This is a transcript of that conversation.

Simon Stender Boisen
CTO at Lix

Can you tell me a little bit about what Lix does?

Lix is a study platform mainly targeted at university students. First, students can buy their books much cheaper than they would be able to get them in a traditional shop. Secondly, they can collaborate with each other as they study the material.

We started Lix in Denmark around three years ago, but now we are also in the United Kingdom, Holland, Norway, and Sweden. And we are expanding into the US market.

Can you tell me a little bit about your role at the company?

I'm the CTO at Lix, I've been here since the beginning. I was brought in by Camilla Lastein, our CEO, to build the platform after some unsuccessful attempts to outsource application development. Interestingly, Camilla thought of the idea for Lix while she was herself in college. Experiencing the pain of buying textbooks the traditional way, she thought, "Why hasn't anyone done this?" The rest is history as they say.

How are you using containers at Lix?

A lot of our platform runs inside containers. As you can imagine, we have a bunch of different services that make up the Lix platform. For instance, we have a service that takes books from the publisher and converts them into an internal format. And then we have our main backend, which does all the business logic.

We use containers to run these services because they make it much easier to deploy and operate the software. You get a consistent way to deploy everything, to monitor everything. That is a big advantage for us because we don't really have an Ops team in the traditional sense. We have a few of our core team, including me, who are doing the Ops as well as development.

Containers are great because the Kubernetes platform takes care of a lot of the ops work for us, and we get time back to improve the software running on the platform, not fixing the platform.

Containers are great because the Kubernetes platform takes care of a lot of the ops work for us, and we get time back to improve the software running on the platform, not fixing the platform."

As far as hosting goes, we run everything on Packet.net, which is a bare metal cloud. I've rarely seen a hosting experience that is as smooth as the experience you get on Packet. You can commission bare metal servers without actually getting in contact with anyone, and you can take them down just as easily. We're really happy with it and it's a great fit for Kubernetes because you don't really need the virtual machines as much as you might have used to. You get to cut out the middleman, performance-wise, which is always great.

I've rarely seen a hosting experience that is as smooth as the experience you get on Packet."

Can you talk a little bit about some of the challenges you needed to overcome in order to run stateful services like databases and cues and key-value stores in containers?

We started with containers three years ago, before any orchestration really existed. But without orchestration, we stopped using containers because we didn't feel it was worth the hassle at that time. Then Kubernetes came around, and we started going into containers again around a year ago. Things worked well, but we definitely felt the hassle of running stateful services and we even discussed whether we should run stateful services in containers at all. With Portworx, when there is a failure we can just say "Okay, we have these six machines, we have Cassandra running on three of them, if one of these nodes goes down, we can actually spin up Casandra over here." We recover much faster from failure than if we were relying on Cassandra to bootstrap that node.

With Portworx, when there is a failure we can just say "Okay, we have these six machines, we have Cassandra running on three of them, if one of these nodes goes down, we can actually spin up Casandra over here." We recover much faster from failure than if we were relying on Cassandra to bootstrap that node."

We decided that we did want the automation benefits of containers for our data services, but there were some challenges.

For instance, we started running Cassandra on Kubernetes but we managed it using manual node labels in order for us to select which machines the Cassandra pods should run on. We did the same thing for Postgres and Elasticsearch which powers our book search service. But this was a huge hassle. First, it is really hard to manage node labels manually. It is easy to make a mistake and it's just a lot of stupidly wasted time. Second, when you lose a machine, we didn't have the ability to put that Cassandra pod onto another machine temporarily while the cluster recovered.

I love Portworx block-level replication for this reason. With Portworx, when there is a failure we can just say "Okay, we have these six machines, we have Cassandra running on three of them, if one of these nodes goes down, we can actually spin up Casandra over here." We recover much faster from failure than if we were relying on Cassandra to bootstrap that node. The other thing that I really like about Portworx is how fast it is. We have metrics comparing Cassandra on direct attached storage with and without Portworx and the line doesn't change. That's great because we got all these advantages and there doesn't seem to be any real disadvantage.

The other thing that I really like about Portworx is how fast it is. We have metrics comparing Cassandra on direct attached storage with and without Portworx and the line doesn't change. That's great because we got all these advantages and there doesn't seem to be any real disadvantage."

What advice would you give someone else seriously considering running stateful containers in production?

Don't try to do it yourself, it's not worth the hassle. Unless you only need to manage a single service and are not that concerned about availability, you should really pick a specialized platform for data management. I definitely recommend Portworx for this because it's working great for us and there aren't that many options to run and manage stateful services on Kubernetes. Portworx saves us time. Time is the most important resource in a startup. With Portworx, we minimize the time we spend on things not directly related to making the world's best study platform.

Portworx saves us time. Time is the most important resource in a startup. With Portworx, we minimize the time we spend on things not directly related to making the world's best study platform."

There are a couple of open source options that we considered before picking Portworx but for our data layer, we really need things to work. We don't want to be pulling our hair out because we're using an open source project that isn't fully supported. It is worth investing in Portworx knowing that there is someone we can call if needed.

Another thing that I will add is that Portworx saves us time. Time is the most important resource in a startup. With Portworx, we minimize the time we spend on things not directly related to making the world's best study platform.

LEARN MORE

Portworx is the cloud native storage company that enterprises depend on to reduce the cost and complexity of rapidly deploying containerized applications across multiple clouds and on-prem environments.

With Portworx, you can manage any database or stateful service on any infrastructure using any container scheduler. You get a single data management layer for all of your stateful services, no matter where they run. Portworx thrives in multi-cloud environments.

Contact us to find out how Portworx's unique storage solution can deliver the availability, performance, and features necessary to minimize stateful application operations costs and improve revenue growth -

https://portworx.com/request-a-demo

portworx.com
www.lix.com/en

 @portwx

 linkedin.com/company/portworx