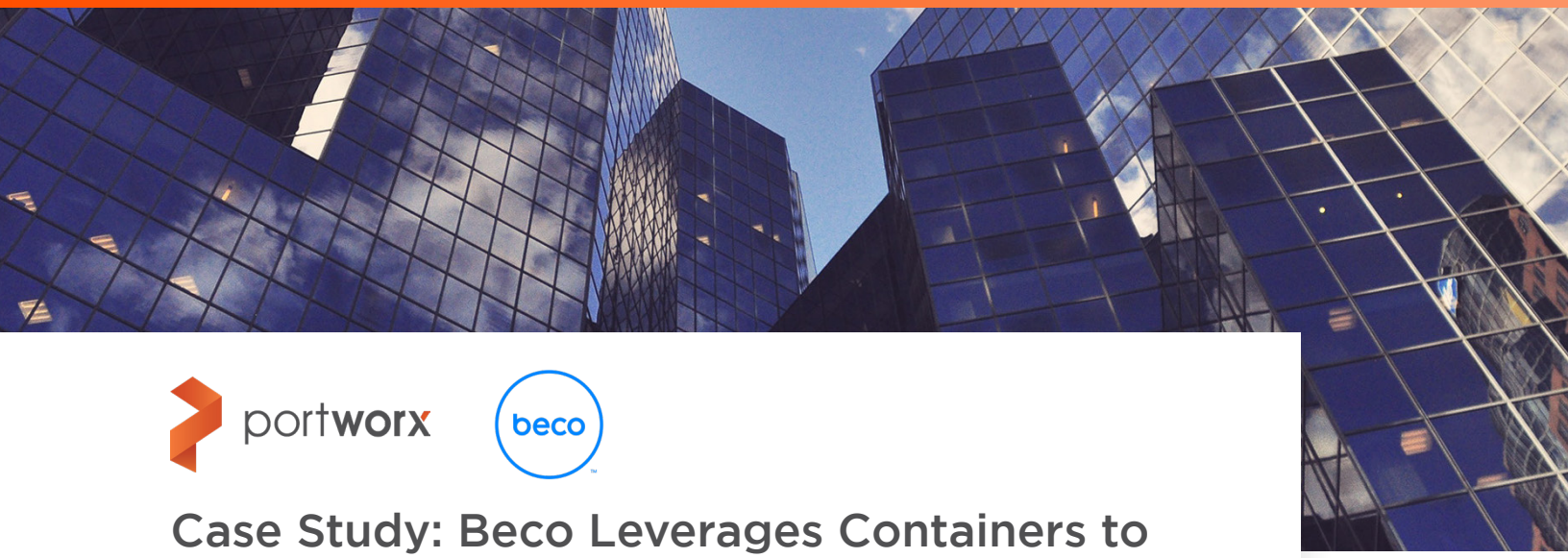

portworx beco

Case Study: Beco Leverages Containers to Manage Mobile Data Stream and Advance IoT in Real Estate

CHALLENGES

- Processing large amounts of data collected from connected devices
- Keeping important databases like Kafka, Cassandra, and PostgreSQL HA when using containers
- Ensuring high availability (HA) when DC/OS pins stateful services to a single host by default

SOLUTION

- Portworx enables databases to run anywhere within a DC/OS cluster
- Synchronous replication keeps databases Highly Available even when a server fails

RESULT

- Beco was able to grow their IoT platform with PX-Enterprise, "An easy-to-deploy, easy-to-operate, and easy-to-scale persistent storage solution for stateful containers"

KEY TECHNOLOGIES DISCUSSED

Container Runtime – Docker
Scheduler – Mesosphere DC/OS
Stateful Services – Cassandra, Kafka, PostgreSQL, WordPress
Infrastructure Provider – Microsoft Azure

We spoke with Jeffrey Zampieron to learn about how Beco is taking advantage of a microservices architecture and stateful services in their business. This is a transcript of that conversation.

Jeffrey Zampieron
CTO at Beco, Inc.

What does Beco do?

Beco delivers real-time space analytics to connect your workforce to physical spaces. We do this by measuring space utilization and workplace collaboration, while activating location-aware features to automate daily tasks.

Through our solution, we offer two main pillars of value for our customers. The first is enabling real-time engagement between people and the space they're in, including wayfinding, finding free meeting space, locating co-workers, and similar tasks. In doing so we are taking friction out of the work environment and helping our customers deliver exceptional work environments.

The second pillar is our long-term analytics. This enables real estate and operations teams to make data-driven decisions about their real estate portfolio and workplace collaboration.

Can you provide some details about how the Beco product works?

Our system is made up of three key elements. The first is our battery-free, solar-powered beacon, which in addition to having onboard sensors, is our "indoor GPS satellite." Deployment of this device is extremely easy and low-lift for our customers. The customer can quickly and easily deploy the system without electricians, IT professionals, or other technical staff.

The second component is our Mobile SDK. Our Mobile SDK embeds directly into corporate and partner mobile Apps to create a data stream on where the smartphone is located in the building in order to deliver location-based services and generate space analytics.

The third component is our Cloud-Service. At the scale we are operating, we are generating a tremendous amount of data via our Mobile SDK that is ultimately aggregated and presented for retrieval, storage, even in real time — and that's where Portworx and Mesosphere come in.

We hold a number of patents with additional patents pending on the technology.

When did you begin with Beco, and what is your role at the company?

I joined Beco shortly after our CEO Tom Zampini started the business, so I've been here since the beginning as a member of the founding team. My responsibilities include design and management of cloud architecture, software development, and advanced algorithms that drive our real-time analytics.

How is Beco using containers generally?

We run a containerized microservices infrastructure that is orchestrated using Mesosphere DC/OS. The container orchestrator within DC/OS is Marathon. Therefore, in my opinion, one way to think about DC/OS is as a highly productized combination of Apache Mesos and Marathon.

There is a lot of discussion around stateless architectures when it comes to microservices, however no matter what direction you choose, you'll need some sort of persistent store. Those can be NoSQL stores like Cassandra, where they're pre-sharded on their own, or they can be classic SQL stores like Postgres or MySQL.

What are some of the challenges you face with container technologies?

When running in containers, most of the orchestration frameworks handle persistence, however they don't do it in an optimal way. The simplest way is that they bind the container to a given host, which takes away from the whole idea of a flexible scheduled container infrastructure by introducing that host as a single point of failure (i.e., if that VM goes down, you lose your volume).

By implementing Portworx, we've gained the advantage that Portworx abstracts the storage hardware and provides not only the ability for your containers to float from node to node for both scale and reliability, but also the data durability that comes with multi-host block-level replication. That has been a missing piece in Kubernetes and DC/OS that you see coming off the shelf.

Can you talk at a high level about the overall application stack and how you're running it in containers?

There are a number of "out of the box" IoT solutions available today that are optimized for internet connected devices, either via a gateway or direct connection – think Nest Thermostat or similar. Our architecture is actually quite a bit different from that, in that we are crowdsourcing our data through mobile, which requires us to actively maintain the location and state of these devices and therefore we created our own high performance data ingestion and stream processing engine.

One of the big perks of containerization is that it handles both polyglot languages and polyglot persistence. It gives you the flexibility to use the right tool for the job, offering the opportunity to use Java, Go, Scala, Ruby on Rails, Django, Web Play, or similar. Using containers offers the flexibility and control necessary to choose the right web framework and language for the microservice being developed.

Could you provide some tales from the trenches about some specific problems you've run into and how you've been able to resolve them?

Container orchestration is clearly on the cutting-edge of technology and as expected, from time to time you run into limitations or features that are yet to be developed. Perhaps the one that sticks out most in my mind was the inability to attach external disks to VMs in a VM scale-set on certain cloud providers. This essentially means that all your stateful containers are node-locked and become single points of failure. Portworx, using standard Linux mechanisms, allows us to attach block devices to our stateful containers in a performant and failure-resilient manner.

Knowing everything you know now, what advice would you give to someone who's starting their stateful container journey?

Handling stateful services in a reliable, scalable fashion is difficult. Portworx provides an easy-to-deploy, easy-to-operate, and easy-to-scale persistent storage solution for stateful containers.

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 nfrastructure 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.beco.io

@portwx
linkedin.com/company/portworx

portworx
4940 El Camino Real, Ste 200, Los Altos, CA 94022
Tel: 650-241-3222 | info@portworx.com | portworx.com