avassa

Confessions of a Platform Engineer, Edge Computing Rollout Edition

01

Introduction

We recently completed an IT-infrastructure project to design and launch an edge platform infrastructure for the company I work for – a company that manages movie theaters.

Like many large IT projects we brought big ambitions. But equally, many large IT projects come with a few honest mistakes, which we definitely made.

Mistakes bring insight, so every downturn had its share of learning, which was good. But after the project was complete, we put together this document – a detailed account of the major milestones – documenting the ups and downs and, honestly, the information we wished we had going into it.

Fear not! Most of our thinking is horizontal and applies to many different types of edge environments.

This write-up covers the first twelve months of our edge computing platform roll-out, scoping all the way to when we started scaling workloads globally. The roll-out took us from our manual, costly, time-consuming, and innovation-prohibitive management process of in-theater applications to a robust platform that our application teams love and will have in place long-term.

*Any resemblance between this book and a certain burning bird is a mere coincidence. Pay no attention to it.

MOVIE THEATER CORPORATION

Our Company

The company I work for is called Movie Theater Corporation.

We're a global movie theater chain with more than 750 theaters in 30 different countries.

The last few years have been a bit topsy-turvy as we — like other companies with a core business dependent on physical locations — need to adapt to a new digital landscape with the visitor experience at the center of it. This has made our marketing and sales departments put pressure on us (and on the application teams). As a result, we need to utilize innovative applications in our theaters if we want to survive.

Our production environment consists of movie theaters as well as two central clouds where we host our back-end systems.

Modern movie theaters rely heavily on software for many things: alarm and temperature systems, high-end soda dispensers, theater room management (curtains?), and digital rights management. Each of these applications needs storage, networking, and compute power to execute. The applications we had running in our theaters when I started working for Movie Theater Corp. were predominately legacy software running on Windows. But we needed to refresh both our hardware and software in order to turn our theaters, as well as our online presence, into one omnichannel digital platform. The time had come to migrate our applications into software containers. And guess who was the lucky person tasked with doing this?

What I refer to as 'edge' throughout this write-up, is really our movie theaters.

My Role

You guessed it. The job fell to me, the lead platform engineer.

Together with my team, I set out to create a platform for managing the applications running in our theaters. In an attempt to avoid siloed teams leading to stove-piped solutions and processes, our IT manager decided to keep the whole platform team as one.

My team (IT and platform) strives toward being enablers for the application teams. In this project, we had a chance to truly exceed the expectations of the application teams as we aimed to create a cloud-like experience for application operations in our theaters. After all, we have a mutual interest in making application orchestration efficient, secure, and robust whether the application is running in a centralized data center, in the check-out line of the ticket office, or at the loading dock of the theater building.

While our cooperation sometimes has its ups and downs, our most important peers in the organization are the application teams.

Our CIO sometimes refers to the application teams as "the value-streamers" to really accentuate where the value is created in our organization. And she's not wrong. The applications are what create a unique, innovative, and comfortable experience for visitors and employees in our theaters.

Our application teams love and cherish the developer experience from the cloud and would like to recreate it as much as possible also when developing in-theater applications. So that, right there, was our goal.

Key takeaways:

Platform/IT teams, and application teams!

Remember the value of working together across organizational boundaries and respecting the wants and needs of both.

The value is created by the application layer and the edge is no exception.

The surroundings of an edge project are unique. This includes how to take on security, distributed deployments, monitoring and observability of applications and infrastructure — and how to scale such an environment using a platform.

Table of Contents

02 Setting goals and getting started

Historically, all software in our theatres was part of products delivered by our vendors that had been locked into their hardware platforms.

Thing is, each of them had their own unique operations stack on which to deploy software and monitor its health. This created an environment that was heavily siloed and cost a small fortune to operate. And it didn't let our application teams reuse any of their application lifecycle tools they had for the cloud.

Directions from leadership

Our strategic leadership was looking for a more agile way for application teams to deliver new features using existing tools and a way to test brand new digital solutions in the theaters without tedious, manual, and time-consuming efforts to truck-roll new hardware.

Our CTO wanted us to think about our theater locations like digital platforms, where application teams could manage applications in the theatres much easier and at a lower cost. They also wanted IT/platform teams to be able to centrally manage the lifecycle of the infrastructure in a fully automated and secure way.

Platform + application = the perfect team

In order to give our leadership team what they wanted, we had to work closely with our application team.

If the new system had to give both the IT/platform and application development and operations what they wanted, there was no way around it. We had to think as one.

Together with the application team, we put together a requirements specification to get started. We did this with the understanding that nothing was set in stone – we might iterate tons of times on the requirements as we built the solution. As long as we were learning and heading in the right (and same) direction, that was good. But with any new project, you have to start somewhere.

Getting the ball rolling

For that reason, we decided to start with a small lab trial with the constituent parts in place.

We wanted to get a feel for how the interactions between our existing teams, the operational tools, and the platform would work. If all went smoothly, we would then go ahead and roll out the platform to a few of our chosen theaters along with a first, production-grade application. Then, the fun would really get started.

Together, the two teams told the strategic leadership about our approach and initial architecture. Thankfully, they approved of our method, gave us the green light, and we could get started.

“

The edge environment will be key in both the digitalization of the in-theater experience and in our efforts to create an omnichannel strategy utilizing both our customer’s in-theater interactions and their digital ones.

CIO

”

03

Piloting in the lab

After the initial planning, it was finally time to get real and set up the physical infrastructure in our lab site.

We have a local test-theater that is a replica of our medium-sized theaters in terms of infrastructure and we use it for all lab activities. (Sadly, our test theater is only that – no movie screens or popcorn machines for us workers.)

Wiring up the lab

Physically installing infrastructure takes time, and that was no different at our test site.

We settled for clusters comprised of three edge computers as this reflected the compute needs of our medium-sized theaters.

Three-node clusters allowed us to provide enough fail-over redundancy to meet our SLAs. It covered outage-style scenarios, and was also a convenient setup for doing node-by-node upgrades on the infrastructure layer (operating system, etc).

And we decided to keep it simple and as real-life as possible, using the tools we normally use in our private datacenters (i.e., networked installation using PXE and tooling from our favourite Linux vendor to keep the OS up to date).

Remote management of infrastructure is different

The installation of the first computers went off without a hitch.

But that's not a surprise – we're pretty good at that. Still, throughout the process, we tried to identify which of the steps could work remotely. This was something we had to do if we wanted this project to work, as we're talking about hundreds of locations with edge computers behind NATs and firewalls. If we tried to do maintenance on site, by hand, it would take forever.

But as we went about this, we realized we had to start looking for a more specific solution that would allow us to manage huge numbers of remote hosts and make that part of our operational capabilities.

Attaching the pipeline to deploy an application

Once we had the plumbing and the infrastructure set up, we huddled with the application team and see how we could connect with their tooling.

We decided to try a simple application that tracks the number of people entering a theater using a camera feed and a simple AI-based application.

In our first try, we looked at the steps it takes to manually deploy the application.

Pretty soon, we integrated our CI/CD pipeline, which gave us a fully automated flow from the build steps to our lab cluster. We had to ensure the current continuous deployment (CD) setup could be extended to include enough configuration to do edge-specific things, like deploy to specific locations.

With some minimal adjustments we were able to allow for declarative definitions of not only what the applications look like, but where (and under which circumstances) they should be deployed.

The close collaboration with the application team was great, and we quickly leveraged their previous work to get things done.

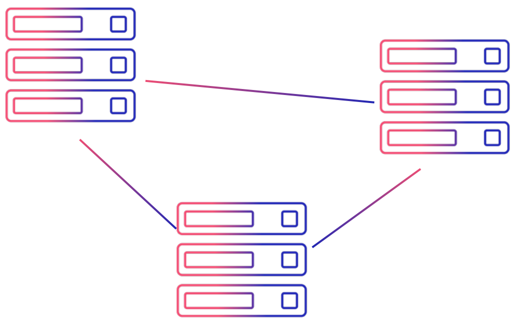
The close collaboration with the application team was great, and we quickly leveraged their previous work to get things done.

Now that the application team has a fully automated release process, we can connect the platform directly to that. And voilà! An edge, as automated as the cloud.

IT Team leader

04 Going into production

Now we were ready to bring our pilot to a small number (25) of production sites.



We replicated the setup from the pilot (three edge computers in each theater), got them wired up, and started on our first production installs.

Installing clusters across the theaters

Scaling the infrastructure fleet management solution that we chose was definitely a learning experience.

Thankfully, with some small adjustments, we managed to get the clusters installed across the theaters supported by automation. Unfortunately, this process exposed a huge grey area around who does what across our organization. But regardless of our internal dirty laundry, we still needed a streamlined large-scale rollout of our infrastructure and edge platform.

The infrastructure team had already wired up the network, installed Linux, and made sure that networking was in place.

By leveraging a system with integrated call-home features we were able to bring up the site clusters in all theaters in a couple of minutes per edge computer. We were relieved, as this could have been a multi-week project if we'd had to physically install the media in each and every theater.

Deploying the applications

We noticed that as the number of sites grew, the nature of the deployment changed. Our very first (and *very* naïve) pilot deployment to a single theater did not work well, as the number of deployable theaters grew.

We were aware that not all applications needed to run in every location, but the application setup in each site was based on various standards like site size (number of theater rooms, size of snack bar, etc.). We needed to tailor the deployment to support these parameters. Once that was established, we could assign responsibility, decide which applications should run where, and ensure this was all done in a controlled, trackable manner.

We split the definition of the application itself and the definition of where the application should run.

By doing this, we avoided lengthy meetings and the creation of a ridiculous amount of manual steps. This alignment allowed us to rely instead on the application team's existing GitOps approach, which not only saved time, but also delighted our CTO as tooling we'd already invested in for the cloud could be leveraged again at the edge.

Monitoring the applications

After the deployment, we still lacked insight into the application health.

Deploying an application to the edge is kind of like throwing the application over a wall and hoping for the best unless you have observability capabilities to monitor the health of a specific application in a specific theater. Luckily, we did.

That sort of application health monitoring across many locations requires collaboration with the operations teams to ensure a coherent view of where applications are running. It also allows us to use data we gain from the monitoring also in integration with existing tools.

05 Our first live infrastructure upgrade

We were pretty thrilled to see the applications running in production for a couple of weeks *and* performing to expectations.

We held off on maintenance (upgrades, patches) during this period in order to observe a stable environment. But once that was set, we needed to do our first infrastructure update in production.

Upgrading the infrastructure in flight

Updates for the operating system and supporting applications (including the container runtime) had been piling up during our change freeze.

We had to start working through the list, no matter how long it was. Using our infrastructure fleet management solution, we brought the operating system and system-level applications up to the appropriate version.

We were careful to tease apart the management of the supporting infrastructure (OS, container runtime) and the running applications and assign responsibility to the platform team and the application team respectively.

Layers interacting

As part of the application deployment, we configured synthetic probes running at each site that continuously reported the application response times.

This proved to be hugely valuable as we could now see degraded application performance at the sites. This was confirmed by our service desk who received reports of the same issues from local theater employees.

End-user performance of applications usually depends on a number of contributing factors up and down the stack.

With that in mind, we huddled together with the application team to figure out what was going on. Thankfully, the probes pointed to specific containers that had degraded performance so we knew exactly where to dig.

Resolving issues across layers

By correlating logs across the operating system and application layers, we figured out that the performance issues came out of some unexpected interactions between the container runtime and the kernel scheduler that ended up introducing significant delays in response times from our containerized applications.

Key to our successful root-cause analysis was our cross-team collaboration.

It was easy to correlate between the application layer (where we started) and the infrastructure (where we found the issue) since we had made sure to keep references between the applications in terms of which edge computer-specific instances were running on, and which container runtime instance and kernel on the infrastructure layer was leveraged by a specific application. Another factor that shortened the resolution time was that our edge platform gave us direct access to the logs without a manual login.

06

Rolling out to all theaters

Now that we were seven months in, we felt we had made significant and meaningful progress. We had something to celebrate!

With the ability to perform targeted deployments, we were ready to begin the automated rollout of our deployment across 30 countries. The difference in time-to-deployment from our previous manual approach was exhilarating.

We really mastered the process of monitoring and observing the in-theater infrastructure applications, so we were confident we could quickly identify any bumps in the road during the process.

This meant a whole new level of consistency for our colleagues and visitors using the digital services in the theaters. To say we were excited was an understatement.

Even better was this helped us avoid unnecessary and time-consuming involvement from those of us on the platform team in the form of endless meetings.

In addition, we could also confidently let the application team utilize a fully automated deployment process, which enabled us to increase deployment frequency even more.

07 Piling on applications

After patting ourselves on the back for the successful deployment of our first in-house application at a global scale, we got back to it.

First challenge tackled, we needed to onboard a third-party party application vendor to supply us with an AI video analytics application to help determine which movie theaters needed additional cleaning services.

Packaging the first third-party application

For this second application, the biggest difference was that we didn't develop the code in-house.

Instead, we took tested releases from a vendor according to a release schedule. We then packaged the application to optimize how it ran on our infrastructure and ensured stellar monitoring and observation.

Since the application was fully containerized we knew that merely scheduling, starting, and stopping it would be easy as pie. But the application needed a secure way to distribute sensitive data (application credentials) to the edge video application from a central location. To do this, we had to have distributed secrets management features.

The need for multi-tenancy arises

In addition, we knew we couldn't let the camera application write to storage in an uncontrolled way, as that might affect (starve out) other applications running on the same cluster. For data protection purposes, we needed the application to be kept separate from our in-house application for data protection purposes.

As a result, we decided to run this second application as a separate tenant, with strict resource separation.

This provided deep isolation in terms of enforced resource limits, network traffic separation, and tenant specific encryption against data extraction.

“

You've got to think about multi-tenancy and resource constraints for IT running out in the theaters. Also, data protection between applications needs to be seriously considered.

IT Team leader

”

08 The Saturday night security incident

Currently, we manage the lifecycle of two applications through our CI/CD and we have monitoring across infrastructure and application layers.

Guess it was the perfect time for a security incident…

Stolen hardware

Just our luck – we get a call on Saturday night from a movie theater reporting the theft of two edge computers after hours.

The computers stored sensitive data including application credentials and keying material. Our main concern was the potential extraction of sensitive data from the physical hard drives.

That was one of those moments when you get genuinely happy by reading the docs.

Turns out that in our edge platform's secrets component, there was a built-in protection for these kind of events. Talk about a relief. All sensitive data was encrypted and could not be accessed by brute force. We also used a knob to lock down the site completely until the incident had been analyzed. While we still were missing the computers, we could be assured that no information from them could be put in wrong hands.

> **Remember to study carefully which data is distributed to the theaters and how we protect that using the platform features. If we wouldn't have routines in place for physical theft it wouldn't have ended well, since perimeter security is non-existent.**
>
> IT Team leader

09 Extended cloud capabilities and ability to scale

At long last, we have a platform that enables our application team to deploy edge on-prem applications.

Many of these applications have a counterpart in the central cloud. We worked on a unified pipeline so that the last deployment step manages central versus edge deployments without negatively impacting the developer experience. We also bridge underpinning API services for things like storage, pub/sub bus, and secrets across central applications and edge applications. This was necessary since by nature, different components are used for the constrained edge versus the resource-rich central cloud.

Previously we had two separate teams, one for cloud and one for edge.

Now that these two teams are united into a holistic application team covering the organization's needs, we were able to unify edge and central application monitoring. And we've never looked back.

5 considerations for running edge pilots

In retrospect, we succeeded with several aspects of the project, while there are a few things we could've done differently. During our next iteration, we will keep the following considerations in mind:

1. **What are the characteristics of the edge environment?** Remember that when piloting, it's important that you know the characteristics of your edge sites so they can be mimicked in the pilot environment.
2. **Setup relevant goals.** When working "from the bottom up", it's challenging to prove value and benefits early on. Therefore, it's important to set up relevant goals that align with the nature of the project roll-out.
3. **Use-case-driven or platform-driven approach?** We felt pressure from application teams to see applications deployed, without really caring about any platforms. Meanwhile, platform teams want to see generic functions, no matter the application. This must be better balanced, and solid collaboration across teams is key.
4. **Consider the complete stack and associated lifecycles.** The edge has several layers: local infrastructure, hosts, OS, container runtime, edge cluster, container applications etc. All layers must be considered in order not to avoid any unpleasant surprises.
5. **Wrap up and evaluate the pilot.** Reporting lessons learned and achievements throughout the organization are key. This will be easy as pie if considerations 1 and 2 are carefully considered.