Rent Like A Champion:

An Interview with CIO, Mike Hostetler

RENT LIKE A CHAMPION

servercentral®

servercentral®

RENT LIKE A CHAMPION
CASE STUDY

COMPANY
Rent Like A Champion
www.rentlikeachampion.com

INDUSTRY
Vacation Rentals

LOCATION
Chicago, IL

CUSTOMER SINCE
2015

Rent Like A Champion

RentLikeAChampion.com connects travelers with homeowners in college towns across the country. The company was started by three Notre Dame Alums who were frustrated with staying in dingy, overpriced hotel rooms for Notre Dame football games that were often booked a year in advance. They thought there was a better way to enjoy a football weekend with family and friends, so they set up a website for homeowners to rent out their homes for the weekend.

Over the past 9 years, Rent Like A Champion has expanded to more than 20 college towns across the country.

AN INTERVIEW WITH MIKE HOSTETLER, CIO, RENT LIKE A CHAMPION

Q: WHO ARE YOU?

Mike Hostetler, Entrepreneur & Engineer, based in Chicago. I exited my first company earlier this year and now work with early-stage startups in various roles helping them scale their technology and business. For Rent Like A Champion, I served the role of CIO and Sys Admin, prepping the site and application to withstand the onslaught of traffic expected from the company's appearance on Shark Tank.

Q: HOW DID YOU COME TO WORK WITH RENT LIKE A CHAMPION?

Jordan Curnes, one of the founders of Rent Like A Champion, is a friend. He graciously helped introduce me to the Chicago business and technology scene. When I finally had some availability this year, I called him asking if there was anything I could do to return the favor. Rent Like A Champion was returning the favor! I originally joined the team to facilitate a responsive redesign of http://rentlikeachampion.com. We finished that project up in May 2015 and immediately learned about the Shark Tank opportunity. As that solidified, we continued working together and realized that the technology stack needed some attention. We had to be sure the application and infrastructure held up under the unusual circumstances of airing on a popular TV show with a national audience.

Q: WHAT WAS RENT LIKE A CHAMPION FACING, TECHNOLOGICALLY, HEADING INTO SHARK TANK?

Going into Shark Tank, Rent Like A Champion was on solid footing for where they were at as a company. The Shark Tank opportunity is unique because of the potential for such incredibly high load over a short period of time. This spike, or "traffic profile" as we called it, was totally unique. Preparing to manage this event directed every planning step we took at that point.

The unique thing about the Shark Tank hit was that it was incredibly difficult to prepare for. First, you don't really know how much traffic will be generated because the number of variables at play is very large. We learned that the show reached 7-8 million people an episode, but didn't have a feel for how many of those people would make the jump to go and visit the website. We knew the outcome of the deal, but didn't know how the segment would be edited or how long it would last. We didn't know if our segment would air early in the hour or late in the hour, which would have an impact on traffic.

> *This meant we had to prepare for the worst case scenario and have it on the shelf ready to execute, potentially weeks in advance.*
>
> Mike Hostetler
> CIO and Sys Admin
> Rent Like A Champion

The last element of unknown was that we didn't receive the official air date until two weeks before the show was planned to air. This created a "hurry up and wait" situation which proved incredibly frustrating and difficult to plan for.

Bottom line? We had no idea what to expect or when. This meant we had to prepare for the worst case scenario and have it on the shelf ready to execute, potentially weeks in advance. There are very real costs associated with this that had to be understood and managed. It was certainly not a low-stress time for the business or the technology.

We were fortunate in that we were able to call other Shark Tank companies and ask for their advice. They were all very generous and shared their numbers with us. However, Rent Like A Champion's application is a modern, fully customized web application, whereas most of the other companies we spoke with were more straightforward marketing and pre-interested eCommerce sites. This is a HUGE difference in complexity due to the dynamic nature of our application.

The last element was that we had one chance to get this right. I called it the Space Shuttle Project because there were no do-overs. You either nail it or you don't, and it will be highly publicized if you don't. This guided a lot of our preparation as well, as every I had to be dotted, every T crossed, and the entire environment had to be stress-tested many times over.

Q: WHAT DID YOU SEE AS THE BIGGEST CHALLENGE TO SOLVE IN PREPARATION FOR THIS TYPE OF UNPREDICTABLE SCALE?

There was just one chance to get it right!

We knew this, but it became more and more evident - and important - as time passed.

As we addressed this challenge, I knew I was playing quarterback and surrounded myself with a team of people who could help me block and tackle at every level. Each layer of our infrastructure was carefully planned, configured, deployed, and alerted to what we expected to happen. We worked extensively with ServerCentral's team to make sure everything was prepped for the challenge we faced with this traffic profile.

A second challenge, or opportunity, really, was caching. Caching proved to be our secret weapon. The Rent Like A Champion application is built on Rails, which is notorious for its inability to scale as a platform. We quickly realized that if we ran Rails without caching, we would need over 100 servers to meet the risk profile we were comfortable with. This was unrealistic from a cost and a technology perspective.

Q: HOW DID YOU ADDRESS THIS CHALLENGE?

The guys at ServerCentral suggested we look very closely at Varnish. Varnish introduced an order of magnitude of complexity into our application, but was our only realistic option to handle such a high load. They introduced us to some trusted resources and tools and we were able to integrate Varnish into our technology stack.

We did run a few tests using nginx (our web server of choice) as the reverse proxy layer, but that proved to be less performant than Varnish. At this point we knew Varnish was going to be more complex, but we knew it was the right answer.

We learned a few things along the way:

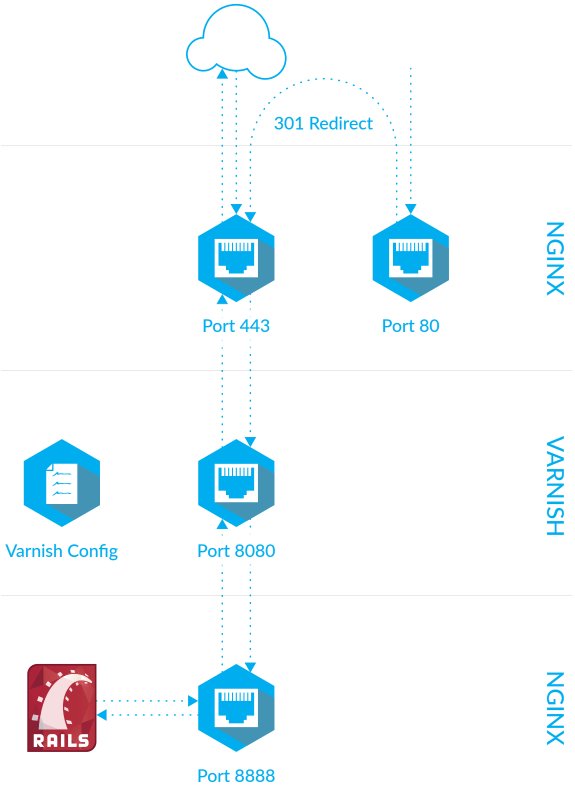
Varnish can't terminate a SSL connection so we had to leave nginx to do that. With our horizontal scaling strategy, I wanted to keep our application servers all self-contained, meaning we didn't need to coordinate another layer into our infrastructure.

> *I called it the Space Shuttle Project because there were no do-overs. You either nail it or you don't, and it will be highly publicized if you don't.*
>
> Mike Hostetler
> CIO and Sys Admin
> Rent Like A Champion

I specifically didn't want to introduce another layer for two technical reasons.

1. We were architecting for a specific event and planned to tear it down the next day. It made perfect sense to follow the KISS rule, keep it simple, stupid.
2. We intentionally chose not to use a configuration management layer like Puppet or Chef because it was overkill for our situation. Ultimately this limited our ability to sustainably build a more reliable and robust infrastructure, but the project objective was to survive Shark Tank.

The solution was to make a nginx and Varnish sandwich on each box.



1. Port 80 redirects traffic to Port 443
2. 443 terminates the SSL connection and proxies the request to Varnish, which is running on Port 8080
3. Varnish then reached out to nginx running locally on Port 8888 where the Rails application sits
4. Insert meme with a picture of a Varnish sandwich

This approach proved to be robust and scalable and was one of my biggest personal lessons coming out of the project. I will definitely be using this again.

With our application organized into this "lego brick", we proved we could scale horizontally as far as necessary.

This approach limited complexity, minimized potential points of failure and was an all-around better economic decision.

Q: ONCE YOU OPEN THE PROVERBIAL CAN OF WORMS, ALL KINDS OF THINGS CHANGE. NOW THAT THE CAN WAS OPEN, WHAT WAS THE BIGGEST CHALLENGE YOU UNCOVERED?

After we nailed down our plan for a nginx and Varnish sandwich, we installed the Rails application and began testing the application. We immediately hit cache issues for logged in users.

The problem was that by default, Rails sets a session cookie for all users. This cookie wreaks havoc on caching. We had to spend a considerable amount of time testing and re-testing our Varnish configuration to get the right setup for our application with regards to cookie handling. This took pairing with our CTO, who was tweaking the Rails application and HTTP headers, while I was tweaking Varnish. In the end, we got it right and learned a lot about Varnish along the way.

After we sorted the cookies out, we did more load testing and observed something very interesting. The first load test I would do after I had restarted Varnish (and flushed the cache) would deliver terrible results. I conducted most of my tests in groups of three, meaning I'd run the test three times and average the results. Tests 2 & 3 would run without issue. We did more digging and eventually realized that on the first load test run, Varnish was populating the cache. The problem was, while the server was doing the extra processing to populate that page to the cache, other requests would queue up behind it so quickly that they'd all hit their timeouts. This would have delivered a bad experience to users before the server could get caught up. We knew if we didn't solve this traffic "shock wave" problem, we would fail.

The more we looked into this shock wave question, the more we began realizing that any sort of auto-scaling strategy would never have worked for us. Automated provisioning is nice, but it just can't respond fast enough to what we were facing, or for what many others face. In a real world situation, with real consequences for failure, there was no way we were going to trust auto-scaling scripts to handle the load.

With more research and discussion with ServerCentral, we realized that we needed to deal with the traffic shock wave head on.

The solution was to was write a cache-population script. We crawled our entire site and generated a list of URLs across the entire application. We then fed this list of URLs to curl to make a single anonymous requests to all 1500 URLs, thereby populating the cache for those URLs.

After writing this script, results from load testing proved we had solved this issue.

We were ready for Shark Tank.

Q: WHAT HAPPENED WITH SHARK TANK?

The night of Shark Tank was a total blast. The company held a party in a club in downtown Chicago, so we holed up in the back and set up a mini-NOC around a bar table.

We knew that once the episode came on TV, we just needed to hold on and enjoy the ride. We also set up a back channel where we could be online with the technology team from our investors and the ServerCentral team. There was a limit on what we could observe, statistics wise, so we focused on Google Analytics and New Relic. The ServerCentral team was watching their infrastructure monitoring tools so we felt we had our bases covered.

We only had one glitch the whole night. At T-minus 3 minutes, Evan at ServerCentral called out that he was seeing "500 errors" on one of our servers. Because we were all staring at our screens, I was able to instantly log into the server with the problem, diagnose that Varnish had crashed, turn it back on and bring it back into the swarm in about 30 seconds. This was my second favorite moment of the night.

The best part was seeing the traffic climb when Mike and Drew came on TV. I had planned to live-tweet the technology side of the event and it was a total blast calling out the number of users hitting the site as it climbed. At one point, we were adding 150 users a SECOND. I kept calling out the numbers on the back channel to the other guys so they knew what I was seeing. Simultaneously, the noise from the crowd in the club was rising as well as it became obvious that the Rent Like A Champion team was about to receive a deal from the two billionaire sharks on the show.

When we finally peaked, we had seen our simultaneous users increase 10,612.5%. It was so loud I don't think the other people on the back channel could hear me anymore. Absolute chaos. It was great. What was more incredible was that New Relic was still reporting less than 250ms response times for our application, which was well within our target response range.

As the segment ended, traffic slowly began to decline and we knew we had nailed it. It was an amazing feeling.

Q: WHY DID YOU CHOOSE TO WORK WITH SERVERCENTRAL?

When I started on the project, we were hosting with a vendor that claims to provide fanatical support. We never really had any issues, but when the Shark Tank opportunity came along and we were facing a real challenge, we made the decision that fanatical support wasn't good enough. Response times and the ability to solve real problems simply weren't there.

We landed at ServerCentral for a few reasons:

1. ServerCentral has integrated the complete infrastructure stack. They own and manage everything down to the network backbone connections. This means there is someone who knows EXACTLY what is going on at every level. They're watching, managing and mitigating every layer beneath us. That's serious peace of mind to have when selecting a partner, especially with an unknown traffic shock wave heading our way.
2. If I'm building something, a foundation like this means I can start building on the 10th floor with confidence. It's the knowledge that I don't have to think about countless infrastructure elements because ServerCentral has it SOLVED. I spend more time solving the problems in my court, and don't have to wonder or worry about things breaking outside of my cloud because I know ServerCentral is taking care of all of it.
3. ServerCentral is relatable and accessible. They are here in Chicago, right down the street, and we knew they would have our backs as we encountered the unknown of the show's airing and the post-airing "new normal".
4. We knew they had faced similar (and much larger) challenges before and would be a helpful and supportive partner in tackling this Shark Tank appearance and traffic profile we were facing.

Q: ANY FINAL THOUGHTS?

When you're choosing your partners, choose wisely.

It is hard enough building a business when everything is stacked in your favor - let alone when you're fighting against yourself.

Remember, if one piece of the puzzle is working against you, it amplifies itself so quickly that seemingly little things become very real big things.

It's not difficult to select quality partners, just be sure you do it!

servercentral®

111 W. Jackson Blvd.
Suite 1600
Chicago, IL 60604

www.servercentral.com
Toll-Free: (888) 875.4804
Worldwide: +1 (312) 829.1111
sales@servercentral.com