NETAPP ON NETAPP EBOOK

NetApp IT's Journey from Monolith to Microservices

NetApp

NETAPP ON NETAPP EBOOK

Content

Supporting customers at the speed of conversation

By Robert Stumpf
Senior IT Director, Enterprise Solutions

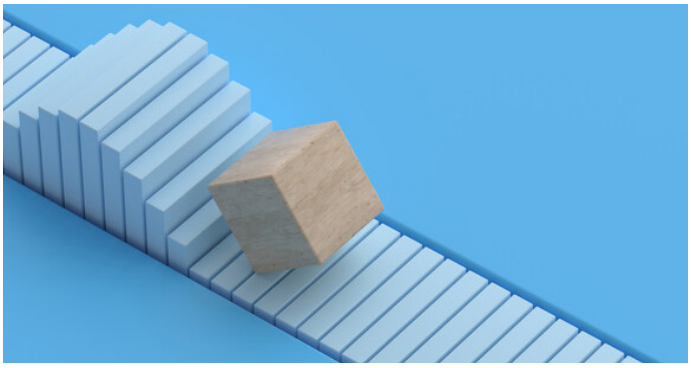
Every day, hundreds of NetApp technical support engineers troubleshoot customer cases on the front lines. They are superheroes who handle 15,000 cases each month. Yet these heroes were using “swivel chair” processes to navigate among nine separate systems to find and capture case data, support bulletins, known bug information, and knowledge base articles. They had to manually retrieve and consolidate this information to provide a solution to our customers.

To capture customer support knowledge in real time and produce technical solution documentation, my solution delivery team—working with our Customer Support Delivery business partners and Service Management peers—rewrote the case management system. We built the new system by using our new DevOps framework, CloudOne Business Resilient Applications (COBRA). Leveraging our internal platform called CloudOne, we built COBRA to provide the templates, libraries, methodologies, and processes to manage cloud-based microservices-based architectures.

We found that treating every microservice as its own software was a key first step, and although each provides a separate service, they all must work together. Microservices scale better and are easier to manage and test. They allow us to optimize resources and have multiple teams work on independent services, enabling us to deploy more quickly—and pivot more easily when needed. As a result, our work velocity is three to four times faster; code deployment happens in seconds, not hours; and bug fixes can happen almost immediately.

A solid UI design was also important to the success of this project. With COBRA and continuous integration and continuous deployment (CI/CD) capabilities built on top of CloudOne, the team was able to quickly tune UIs in response to user feedback. The application was rolled out in phases with weekly releases to iterate on improvements based on user feedback.

The new case management system has nine microservices that retrieve data from SAP, NetApp® Active IQ®, troubleshooting tools, and knowledge base systems. All data is delivered through a custom-designed UI to eliminate swivel chair processes and increase the TSE team's productivity.



Implement Knowledge-Centered Service

One of the project's objectives was to help our Technical Support Center transform by implementing an industry-leading Knowledge-Centered Service (KCS) v6 practice. KCS methodologies increase the speed of creating and publishing knowledge to customer self-service, reduce the time to proficiency for new TSEs, and decrease time to resolution for cases with known problems. Specifically, we integrated our knowledge management and search systems into the support engineer interface so that TSEs could apply KCS service delivery methodologies. This kind of integration was not previously possible.

A Smarter, Faster Support Center

Today, our TSEs can engage in real-time knowledge capture that moves at the speed of conversation with our customers. The NetApp Support

Center has increased its rate of adding or updating knowledge tenfold, contributing over 1,500 adds and 5,000 changes per week. The system has many embedded data validation points that proactively look for proper conditions. It validates situations like ordering a part that doesn't fit the customer's installed base or processing a customer case that isn't entitled to support. Another benefit is that it's easier to find customer-specific instructions for on-site part deliveries and field service, helping customers with limited data center staff.

The new system has enabled automated case creation based on inbound caller data. When a call comes into the Tech Support Center, the system uses an algorithm that automatically enters information from the telephony system to aid call routing or populate the initial ticket. All this has improved the productivity of our TSE community.

Lessons Learned

We were fortunate to have an SAP expert on the team who knew how to take full advantage of the SAP feature sets and UI microservices architectures. His expertise and knowledge had a significant impact on our success and ability to make improvements.

We learned to spend extra time on the UI when building an app with very specific designs. Although our DevOps framework made it easy to change the UI, a good first pass was important to address requirements, because we were dealing with massive amounts of data that came from the former spreadsheet dashboards. This approach was especially important in building a system for technical engineers who love information-intensive screens.

It was challenging to rewrite a system for a user base that is technically savvy, highly opinionated, and enamored by details. Although we found the TSEs reticent to give accolades, they did express appreciation of the single data-entry screen, embedded knowledge articles, predefined templates and pulldown menus, and the training videos and instructions on the new system.

10x

The NetApp Support Center has increased their rate of adding or updating knowledge by **tenfold**, contributing over 1,500 adds and 5,000 changes per week.

Rebuilding a monolithic enterprise application with microservices and DevOps

By Florian Lippisch
Senior Manager, Services Enablement Solutions

Like many companies, NetApp runs some old, large enterprise applications that are maintained like zombies. Barely kept alive, these apps are important to the company yet are complicated to keep updated and are expensive to support. As the IT leader for the developers who focus on custom business app development inside NetApp, my team and I know firsthand the challenges of rebuilding customized apps that are founded on monolithic architectures.

Rebuilding Our First Critical Business App

With approximately 350,000 unique visitors per month, the NetApp Support site is one of the most important channels for our customers to get help when they need it. They can use self-help, interactive chat with a support agent, or other forms of digital support. It is an important part of our enterprise landscape and is an award-winning support site.

Still, until recently, the Support site had never undergone a major overhaul, only fragmented enhancements. Certain parts of the site were still running 20-year-old original code. It had become harder and harder for us to maintain and to improve the site. New features were added as bolt-on applications,

resulting in a mixture of technologies and integrations that became risky with the security challenges that are prevalent with older, cobbled-together code.

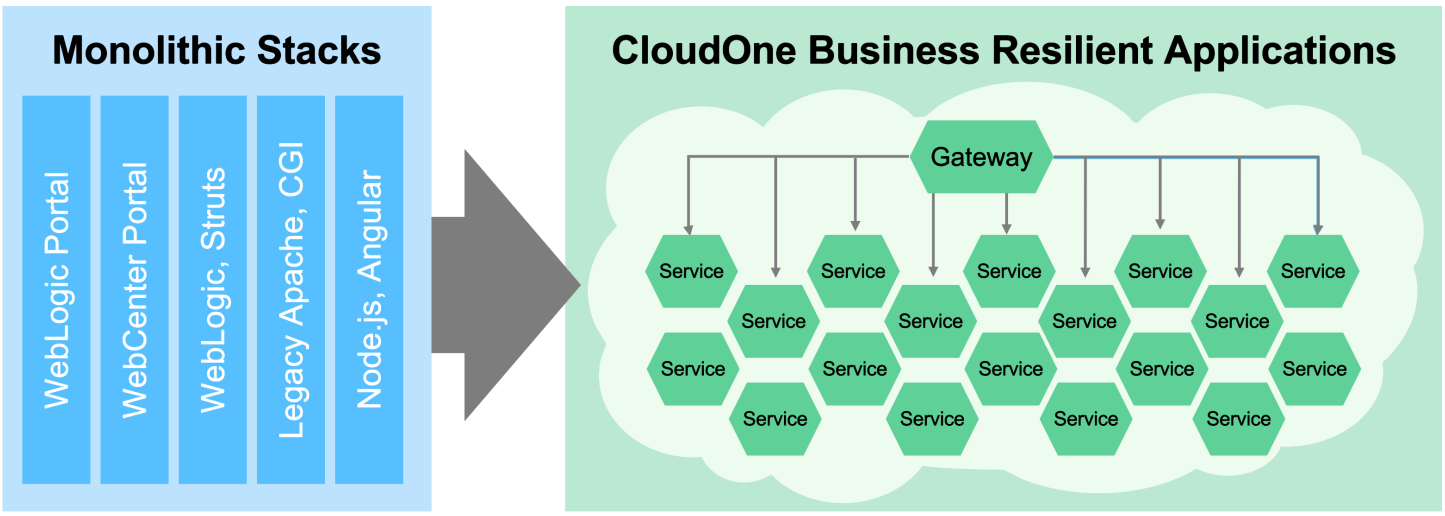
When it became clear that something had to be done with the site, we also recognized that our business models were changing faster than ever and that digital support was of the utmost importance. From an application support perspective, we needed to be able to react to change requests quickly and cost-effectively. The only way to achieve this goal was to rebuild the entire site from scratch.

For the new architecture, we decided to use a microservices-based approach that was developed with DevOps principles. This approach was a great fit for our needs.

As the following figure shows, we went from monolithic stacks to an Angular-based single-page application (SPA) UI that uses microservices running in containers.

By using microservices running in containers, we could break down a complex, large monolith app into smaller, easier-to-manage services. Such services are easier to test automatically through unit and contract-based testing. This testing is a key aspect because it's important to "trust" any changes that are being made. Only then can we roll out updates quickly and with confidence.

The scalability of a container application to handle peak usage periods is another important aspect. Because the Support site is a large public-facing site, we must be prepared to handle peaks without



disruption. Running an application in containers gives us a great abstraction layer from the infrastructure, and it makes the application more fluid. We can now easily move parts, or even the whole application, into the public cloud or back to our private cloud if we want to. Microservices running on containers were clearly the right architecture choice for the rebuild of the site.

Also, the timing is perfect for the Support business team which had just completed a user-centered design to modernize the Support site. The new design optimizes operations around user choice, provides rapid content publishing, and supports low-effort customer experiences. The timing also aligns perfectly with an effort that my coworkers on the IT Foundational (Infrastructure) Services team had just delivered: a new DevOps-based platform called CloudOne.

The CloudOne platform was built to provide all the cloud and other technologies that my developers needed. It starts with a quick initial setup through a self-service portal that triggers automated processes to set everything up end to end. But it does not stop there. It handles all the infrastructure setup through automation as part of our deployment pipelines and provides a rich set of monitoring and management tools.

With all the CloudOne features, for the Support site rebuild, my team could simply focus on developing code to run in the provided containers without having to worry about anything infrastructure related.

With our maiden voyage into the world of microservices and containers, we learned several things.

Early Lessons

- No system lives in isolation, which is especially true for the Support site. Bringing this new, modern world together with our legacy environments was certainly a tough task. And although DevOps is about getting things done with speed, it is not necessarily true for large legacy systems. Starting the integration as early as possible is therefore important.
- DevOps comes not only with technical processes such as continuous integration and continuous deployment (CI/CD), but it also applies strong agile practices on a project that affects everyone. All the speed that you get from the architecture and platform automation requires the inflow of work, the testing, and the feedback loops to be performed in a timely manner. Getting everyone into the right "flow" takes some time and practice across all parties who are involved in the project. And proper change management is vital to get there quickly.

Next Steps

Although rebuilding the NetApp Support site wasn't always easy and required collaboration and support from different levels, it was totally worth the effort and was a great experience for my team. And there's more to the story.

Throughout this ebook, my team and I will share how we determined what technologies to use in our microservices architecture, what choices worked well, and what mistakes were made along the way.

Using microservices to access data from monolithic systems

By Amit Vij
Business Systems SOA and Integrator

The NetApp Support site, which provides our customers with self-help, interactive chat, and other forms of digital support, has been rebuilt on smaller, easier-to-manage services. The site has been transformed from an application with monolithic stacks to a modern microservices-based architecture, front ended by an Angular-based single-page application (SPA) UI.

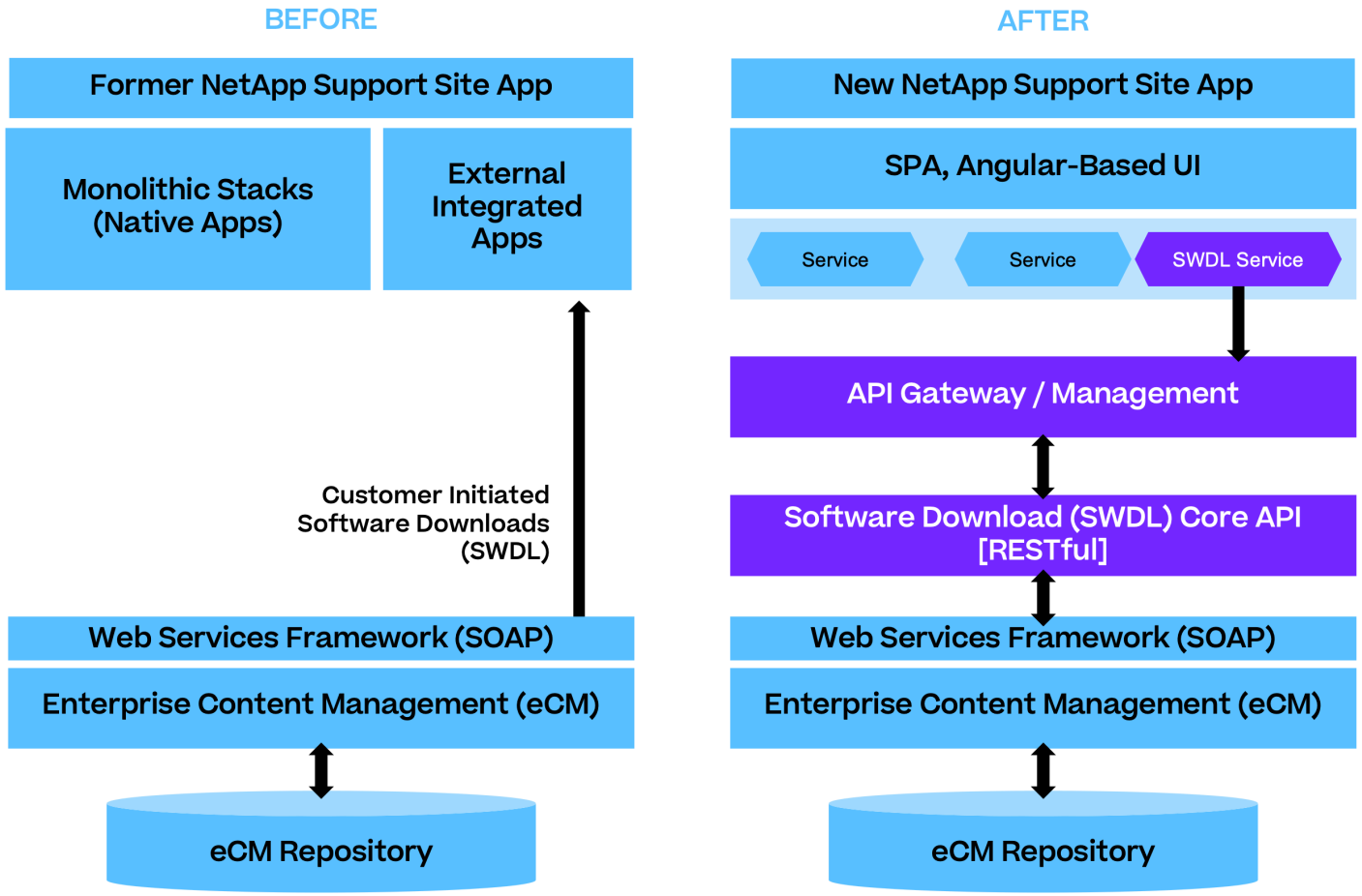
One key back-end system for the Support site is our Enterprise Content Management (eCM) application and its content repository. The team analyzed the functionality associated with the legacy monolith eCM and identified granular services that needed to be exposed. One identified service was the Software Download (SWDL) function in eCM.

An important piece of our transformation was the continued need to retrieve vital data from several back-end systems that would remain monolithic, at least until modernized in the future. Our challenge was to develop services that interface with the legacy back-end monolithic applications. We chose a solution that delivered RESTful APIs, exposing the desired functions, abstracting the legacy systems, and interfacing through an API Gateway platform.

API Abstraction Layer

RESTful APIs were developed to provide the identified services. These RESTful APIs were registered in the API Gateway platform and accessed by the microservices from the service layer. The RESTful API completely abstracts the legacy systems and masks its complexity. The SWDL service can then be reused by other front-end applications that require software download functionality. Other applications that need SWDL function can also access the SWDL APIs directly. The back-end eCM system is completely abstracted from this function.

This new mechanism offers a way to break down our legacy monoliths and convert them into modern microservices-based applications. The legacy system can be transitioned to any modern application framework, providing that it preserves the API signatures, so that consumers can seamlessly access them.



API signatures act like a contract between provider and consumer.

The new NetApp Support site runs on CloudOne, our DevOps-based platform. This platform handles all infrastructure setup through automation as part of our deployment pipeline. It provides all the cloud and other technologies needed via quick initial setup through a self-service portal that triggers automated processes to set up everything, end to end.

Lessons Learned

The first step in our journey was having a robust cloud CI/CD infrastructure; expect a steep learning curve to adapt these new mechanisms.

- A deep understanding of business requirements and an implementation plan are necessary to define the boundaries of microservices; those boundaries are not always clear. In our case, we grouped subfunctions with the business process into a single microservice.
- Compared to a monolithic infrastructure, there are many layers and parts in the new design. Troubleshooting issues could take more time than usual; however, once the teams is familiar with all aspects of the design, the time to market new features is greatly reduced.
- A DevOps approach brings change to people and processes as well as to technology. Successful DevOps adoption requires a consistent push and support from senior management.

Building web-based apps with microservice technologies

By Daniel Otto
Domain Architect, CloudOne Business Resilient Apps

As applications rapidly migrate to public and private clouds, application developers like me are expecting more from traditional enterprise infrastructure. Simply providing servers and storage in a corporate data center, even in a cost-effective way, is no longer adequate. Inside NetApp, our IT Infrastructure team embraced this demand and built a highly automated, self-service, cloud-based application development platform called CloudOne.

CloudOne was built for the rapid onboarding and delivery of integrated software development pipelines and new application run-time environments (for example, containers). To effectively use the platform, we needed an application development framework. As a member of the application development team, I became the domain architect for the new CloudOne Business Resilient Applications (COBRA) framework.

COBRA Framework

We built the COBRA framework as a standard template for implementing web-based applications with microservice technologies (by using Java Spring Boot) in combination with an Angular UI. IT includes automated builds, deployment

processes, orchestration, and monitoring and logging capabilities. We created the framework to build web-based applications, even though it isn't limited to this type of application.

For the new architecture, we decided to use a microservice-based approach that was developed with DevOps principles. This approach was a great fit for our needs. We went from monolithic stacks to an Angular-based single-page application UI that uses microservices running in containers. Technically, we differentiated between four zones:

The **client** is the web or mobile application. The application can be built with any technology that can make HTTP calls to the UI gateway, even though Angular is the preferred client framework for building a single-page application.

The **UI gateway** is the entry point for the client to access back-end services. The UI gateway handles all calls from the client to the services, routing them to the appropriate containers according to the internal configuration. Anonymous calls are allowed for certain routes, but if a user wants to log in, the login is handled by the UI gateway. It also handles user authentication that uses OAuth 2.0. Any identity provider that supports OAuth — for example, Google or GitHub — can be connected. The UI gateway is technically a microservice.

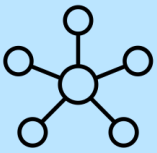
Client

- Untrusted
- Connects via HTTPS to the UI Gateway
- Browser on mobile devices or desktop clients
- Native mobile apps

Microservices Environment

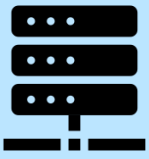
- Accessible from within CloudOne only
- Microservices hosted in container on Kubernetes in CloudOne

UI Gateway



- Accessible from outside (DMZ)
- Allows access for both anonymous and identified users
- Handles identification process
- Hosted in CloudOne

Internal Boundary Systems



- Systems in corp network
- Preferably accessed through axway API Gateway

The **microservice environment** runs in CloudOne and is accessible only through the UI gateway. Each microservice is like a standalone product: It has its own repository and build/deployment pipeline, and it runs as a pod on its own. It's accessed by other services, such as the UI gateway or any other microservice in CloudOne. It isn't directly accessible from outside the environment.When a call is made to a microservice, a security token must be added to the request. The UI gateway does this automatically, even for anonymous users. This action requires services, which call other services, to obtain a security token from the UI gateway before making the call. Although microservices can be built with any technology, we used Java and Spring Boot for our framework. In certain cases, developers may use other technologies if they provide substantial reasons.

Internal boundary systems are our major legacy applications like Oracle Enterprise Resource Planning (ERP) and SAP CRM and are accessible only in the corporate network. The CloudOne environment, though, doesn't allow any direct access to the corporate network, which is why all calls to these systems need to get routed through a specific API gateway. This approach provides an extra level of security.

Last year, we successfully rebuilt the NetApp® Support site by using the COBRA framework and its microservice-based architecture. With approximately 350,000 unique visitors a month, it's NetApp's digital channel for customers and partners who seek information and support related to our products.

Meet the NetApp IT experts

Robert Stumpf is the Senior Director of Enterprise Solutions Delivery and is responsible for all the business applications development and delivery of IT projects at NetApp. In additional to playing a major role in the successful transformation of IT since 2015, Robert and his team were instrumental in the quick and successful replacement of cloud-based Sales Force Automation with SAP Hybrid Cloud Customer (C4C).

Florian Lippisch is the Senior Manager for Services Enablement Solutions and manages a strong and talented team of technical engineers who develop custom applications to support NetApp's global business processes. Florian and team build applications from the ground up—using generic open source frameworks—to implement features not readily found in packaged business applications.

Amit Vij, business systems SOA and integrator for NetApp IT, manages the infrastructure for Oracle WebCenter Content, a multitenant enterprise platform. He designs solutions based on automation, user experience, and content publishing.

Daniel Otto is the Domain Architect for COBRA, the framework inside NetApp for managing cloud-based microservice-based architectures. He is a passionate software engineer for 25 years, living in one of the most beautiful spots in Bavaria, Germany.

For more information on CloudOne and NetApp IT's DevOps Journey, check out the four-part DevOps eBook Series.

NetApp