Executive Summary

Mercedes-Benz Research & Development North America (MBRDNA) sought to enable its distributed innovation teams to move hundreds of microservices to the cloud — thereby advancing the next-generation of connected car services. Multi-cloud support and granting development teams more ownership of the services and applications they deploy were a priority. With Pulumi, the MBRDNA team found the ideal toolset to tame the complexity of many teams and many clouds by making a platform that would enable their vision for the future of automotive transportation.

Learn how Mercedes-Benz builds self-service clouds with Pulumi

About MBRDNA

Mercedes-Benz Research & Development North America (MBRDNA) continuously strives to remain at the forefront of automotive innovation. MBRDNA is headquartered in Sunnyvale, California, with key focus areas of Autonomous Driving, Advanced Interaction Design, Digital User Experience, Machine Learning, Customer Research, and Open Innovation.

Innovating in the Era of the Connected Car

Scaling operations to meet the demands of fully connected cars requires ongoing creativity and agility. Every product and process must be subject to modification at any time, and efficiency and ongoing excellence must be top of mind for everyone in the organization. To support these distributed efforts, in November 2017, the company announced a new Digital Hub based in Seattle, which focuses on developing cloud architectures and platforms for the next generation of connected car services.

“Our core focus is cloud architecture and building out the cloud platform for the next generation of connected car services,” said Dinesh Ramamurthy, Engineering Manager, MBRDNA. “We also have application development teams responsible for connected car-related microservices. We have hundreds of these different microservices that are based on-premises, and we are in the process of migrating them to the cloud.”

Ramamurthy leads a cross-functional team that is responsible for both application and infrastructure architecture. He also works closely with the open innovations team in Sunnyvale. As a result, he wears a lot of hats, and has a lot of people and processes he is striving to optimize. Ramamurthy is constantly on the lookout for new solutions to the multitude of challenges that present themselves in all-cloud environments.

Developers in the Driver’s Seat: Programming the Cloud with Pulumi

Mark Maleitzke is an Engineering Manager and Korey Chapman is a Staff Infrastructure Development Engineer on MBRDNA’s platform team. The team focuses on providing developers with automation and tools that increase development velocity, such as CI/CD pipelines and ways to provision and manage cloud infrastructure. The team is always looking for ways their group can encourage developers to be more agile and independent when it comes to building and deploying cloud applications and services.

Mark says, “Our team runs the core cloud infrastructure that Mercedes uses as the backend to its cloud-connected vehicles. We also provide automation, tooling and pipelines to the development teams that build services for those vehicles. Our goal is to help developers build, maintain and upgrade the infrastructure they need as easily as possible.”

The platform team had a goal of enabling over 200 development teams to independently and quickly provision cloud infrastructure that’s standardized and embeds best practices. This would also help teams still running in the company’s legacy datacenter migrate to the cloud. But with so many teams and over 600 services, the platform team needed a scalable, automated way of providing infrastructure to developers. Furthermore, they needed a way for developers to independently maintain and release changes to their infrastructure after it was provisioned. They knew that Infrastructure as Code (IaC) could help them accomplish both goals, but they needed to select the right platform that would meet their needs and the needs of the developers.

Why Build a Self-Service Infrastructure Platform?

The platform team decided to build a self-service infrastructure platform that would allow developers to quickly create new cloud environments, such as Kubernetes clusters, without relying on the platform team to do it for them. A developer could set up a Kubernetes cluster on Azure just by filling in a few fields in the platform interface. They could also customize the environment type (such as development or production) and what size Kubernetes cluster they needed. The platform would then deploy a Kubernetes cluster with a subscription, subnet, and public-facing IP address. At the end of the process, the developer would have an operational Kubernetes cluster where they can deploy their container.

IInitially, the team built the platform using Terraform as the engine for building, deploying, and managing infrastructure as code. However, they quickly ran into limitations with Terraform and its domain-specific language (DSL), which lacked basic programming constructs, so that it was difficult to model modern architectures. The DSL also made it difficult to manage the additional complexity of deploying Kubernetes applications. Finally, the DSL posed a barrier to adoption for many developers because they would need to learn the language from scratch.

Why Pulumi

The team had been evaluating Pulumi while they were building the platform and Pulumi’s advantages, when compared to Terraform, immediately became clear to them. In particular, they could now use general-purpose languages like Python, TypeScript, Go, and C#/.NET to model infrastructure. Using the full power of programming languages, they could now easily model modern cloud architectures and simplify verbose DSL code using constructs such as functions and reusable modules. Furthermore, they could now use standard software engineering practices and tools to increase development speed, such as unit tests to check for errors or CI/CD pipelines for automated testing.

Pulumi’s multi-cloud and cloud-native support were also important to the team. With Pulumi, they could now model both cloud infrastructure and Kubernetes resources with a common platform and language. Pulumi’s native provider SDKs for Azure and Kubernetes have 100% coverage of each platform’s APIs, along with same-day access to newly released features. Pulumi would also allow them to continuously deploy changes to their clusters with GitOps and integrations with their CI/CD pipeline.

Pulumi also increased adoption of cloud engineering and IaC because developers could now use familiar languages they already know. The platform team created an on-boarding guide that includes Pulumi code examples so that people can quickly get up to speed. They created reusable infrastructure components written in Python that any developer can use or modify. In addition, the self-service platform can export infrastructure code for newly provisioned environments, which developers can use and reference. Pulumi can also import existing infrastructure that was provisioned in another way. In short, Pulumi enables developers to share and reuse infrastructure code just like application code, which encourages collaboration and saves valuable time.

One way the team encourages developers to adopt IaC is to show how, with Pulumi, they can import existing infrastructure that had either been created with another tool or with Pulumi. The team found that being able to give developers access to the infrastructure they already knew was a better approach than making them start from scratch. If developers can use the existing infrastructure, whether they customized it themselves or not, then the code the platform creates for them is close to what they already need.

Mark says, “We've had teams that enter into the IaC world with Pulumi and they’re anxious to know if they’re doing things correctly. Pulumi has been great about bringing the right people to the table to have those conversations and help our people decide if they’re using the right approach.”

The Road Ahead

Mark and Korey see a number of critical uses for Pulumi moving forward. Their short-term goals are to further increase adoption of cloud engineering and to help more teams adopt cloud-native architectures with Pulumi.

They also want to expand their cloud footprint to AWS in addition to Azure. Pulumi provides them an easy way to manage AWS resources with the same languages they're already using, and its AWS Native Provider offers a fine-grained programming model that leverages the AWS Cloud Control API under the hood.

Finally, they are investigating Pulumi features they haven't used yet, such as policy as code, that will help them define and enforce best practices across infrastructure that developers spin up.

How Pulumi Benefits MBRDNA

- The platform team can support hundreds of teams running hundreds of services with self-service cloud infrastructure and ready-to-use Kubernetes clusters.
- The platform team increased IaC adoption by enabling developers to use general-purpose programming languages such as Python, Go, TypeScript, and C#.
- Developers can easily migrate their legacy, on-premise services to Kubernetes or cloud native architectures by using a self-service platform that's powered by Pulumi.
- Developers can leverage and reuse infrastructure code templates that encapsulate architectural best practices for building on Azure and AWS, and all templates are written in general-purpose languages.

Pulumi Corporation

Pulumi's Cloud Engineering Platform unites infrastructure teams, application developers, and compliance teams around a unified software engineering process for delivering modern cloud applications faster and speeding innovation. Pulumi's open-source tools help infrastructure teams tame the cloud's complexity with Modern Infrastructure-as-Code using the world's most popular programming languages and communities, including Python, Node.js (JavaScript, TypeScript), Go, Java, YAML and .NET (C#, F#, VB).

Get started using Pulumi on Azure today