WHITEPAPER

Modernizing Legacy Applications With Microservices

"Microservices present a new design approach that keeps solutions small, modular, and promotes extensibility. These benefits are further enhanced when paired with serverless computing..."

BACKGROUND

In both commercial and government information technology (IT), it is common to see large, monolithic applications grow in size and scope at a rapid pace to a point where the application becomes nearly unmanageable, unsustainable, and unable to adapt to changes in the business environment. The more successful an application is initially, the more it is likely to grow in its capabilities until the application operates well outside of its original planned scope and is burdened by too many features. Many of these solutions are developed as monolithic applications where most features are tightly coupled into a singular environment. In situations like this, the monolithic environment may become an impediment to progress, slowing down the software engineering team's ability to enhance applications and resolve defects. Microservices present a new design approach that keeps solutions small, modular, and promotes extensibility. These benefits are further enhanced when paired with serverless computing which adds tremendous scalability and performance, and drastically reduces the cost of ownership with little effort.

WHITEPAPER

WHAT ARE MICROSERVICES AND SERVERLESS COMPUTING? WHY DO THEY MATTER?

Microservices are discreet, self-contained, easily deployed, and easily managed services that operate on their own but are designed to be integrated into larger solutions. The key to microservices is that they serve a specific, well-defined purpose, typically determined by decomposing business capabilities down to a single verb, noun, or use case. A microservice is built to operate in its own environment and communicate with other microservices using highly efficient and loosely-coupled communications. They are usable by web, desktop, and mobile applications as well as other microservices.

Serverless computing enhances microservices by providing an ideal platform that scales up and down instantly and with little to no interaction. Serverless computing differs from standard cloud web hosting in that the developer does not need to manage the server. The code is deployed to endpoints and the service fabric then completely manages the code. In times of high utilization, the service fabric increases the number of instances to meet performance demands. As the workload decreases, the service fabric automatically decreases the number of instances. Serverless computing is an ideal underpinning for highly modular solutions such as microservices. As a bonus, serverless computing service charges are incurred only while they are running. Combined, microservices architecture and serverless computing provide highly secure, extensible, and scalable environments while also providing tremendous cost efficiency.

HOW ARE THEY DIFFERENT FROM DEVELOPING CODE COMPONENTS?

Many developers attempt to build modular systems using code-level components. These components are dropped into existing applications and compiled and deployed with each application they are used in. On the positive side, this will result in some reuse and will also provide good performance. On the negative side, this approach is not as reusable as microservices and it leads to configuration management challenges where different projects may require different

versions of the component. Once a library is updated, applications that use the components must be updated, recompiled, and redeployed, even if it is for a minor change.

Microservices are self-contained. They are built and compiled as a separate project. They are deployed in their own containers and typically have their own databases. They are not compiled directly into solutions but are standalone services designed to be connected via standards-based, loosely-coupled communications protocols such as Representational State Transfer (REST) and JavaScript Object Notation (JSON). In most cases, microservices are carefully versioned, allowing different applications to continue to operate even if a newer version of the service is released and without necessarily requiring recoding, redeployment, and rework.

For example, a microservice may be developed to provide a catalog of products, another to focus on managing procurements, and another to manage payments. These microservices might be built independently but also provide the core for an end-to-end solution for end-users to browse for products, manage acquisitions, arrange payments, and track progress. When combined, these atomic services can serve many other types of solutions and provide a set of "single truth" per business subject area that can be connected to manage entire business processes. Agile development teams with experience in developing microservice architecture-based solutions will have a keen eye for decomposing solutions into discreet service and connecting them efficiently and extensibility.

APPLY MICROSERVICES AND SERVERLESS BEST PRACTICES

The best approach to getting started with microservices is to identify the right application to modernize that will help establish and capitalize on early successes. The application should be in the right business domain, and with the right contractor to guide your organization through the pilot project. To assure success in the first microservices initiative, the following best practices apply:

- Identify an application that supports a well-understood business domain that is easily decomposed into microservices.

- Identify an application that supports processes that are well defined. Having to wait for business process re-engineering may slow the effort.
- Target applications that already have funding for development or modernization if possible.
- Engage experts in microservices and serverless solution design with demonstrated experience in fielding microservices.
- Build and deploy the solution using leading-edge, serverless design platforms such as Amazon Web Services (AWS) Lambda or Azure Functions.
- Employ compatible development processes, including DevOps and continuous integration/ continuous delivery (CI/CD) to achieve maximum value as early as possible.

By transforming a project leveraging microservice architecture on a serverless computing environment and by building it using experts in DevOps and CI/CD, an organization can see excellent results early and often. The result will be faster delivery of actual value, rapid and consistent deployment of working code, lower cost of operations and maintenance, and a higher degree of reusability.

THE POTENTIAL PITFALLS OF MICROSERVICES

Technology employed by inexperienced personnel can lead to undesirable results and disappointment—microservices is not an exception. A common mistake is not carefully defining the scope and purpose of the microservice, allowing it to grow into its own monolith. Tightly binding the services together and allowing too many dependencies makes microservices brittle, prone to refactoring, and reduces extensibility. Conversely, making microservices overly granular creates microservice sprawl. To combat this, federal IT organizations should enlist contractors with proven experience in decomposing applications into microservices. For maximum benefit, the contractor should also possess a proven background in DevOps, CI/CD, and automated cloud deployment.

CASE STUDIES

eGlobalTech (eGT) is supporting the transformation of a mission-critical, Oracle-heavy system at the Department of Homeland Security (DHS) that is utilized for emergency communications to a

light-weight microservices architecture. We applied the strangler pattern to incrementally refactor existing application code into microservices and deployed to AWS Lambda, a serverless deployment model. The combination of microservices and serverless is enabling eGT to save the customer from expensive Oracle licenses and optimize the cloud expenditure while simplifying operations and maintenance processes.

At Health and Human Services (HHS), our customer was challenged with managing and maintaining more than six systems for a diverse number of grants programs. eGT implemented a new grants performance management platform applying microservices principles replacing existing systems and consolidating all the grants programs into one common platform. By engineering a flexible design, we can declaratively on board new grants programs and changes to existing program requirements with no changes to underlying software. eGT deployed the first production release in less than four months to Microsoft Azure, and since then has continuously delivered new features and capabilities through pipeline automation.

CONCLUSION

Microservices offer a pathway to get under-performing modernization initiatives on the right track. They offer a quicker yield to tangible results due to their highly modular design. By accelerating the development of high-performing, modern applications, microservices break the cyclical, large-scale modernization pattern experienced by most federal agencies every ten or so years. It can facilitate continuous and perpetual evolution of applications and permanently shield them from becoming obsolete and costly. Microservices deployed on modern serverless computing environments drastically enhance scalability and reliability while providing substantial cost avoidance. But achieving these results requires support from agile development teams with demonstrated experience in transforming monolithic applications into a microservices architecture.

Contact us at info@eglobaltech.com to find out how your organization can begin the digital transformation to microservices!

3865 Wilson Blvd, Suite 700 | Arlington, VA 22203 | Office: (703) 652-0991 | Fax: (703) 842-7591 | www.eglobaltech.com