# Portability in the Cloud:

## Best Practices for Building SaaS-Based Applications

# Table of Contents




© 2023 Akamai |

# Cloud-Native Technologies and Architectures

# Cloud-Native Technologies and Architectures

- Microservices
- Containers
- Serverless
- Event-Driven Architecture (EDA)


© 2023 Akamai |

# Microservices

# What are Microservices?

- **Small, independent services:** Microservices are self-contained, modular units that handle a specific function or business capability within a larger system.
- **Scalable:** Microservices can be scaled independently based on demand or performance requirements, making it easier to manage system resources.
- **Decoupled and loosely-coupled:** Microservices communicate via APIs or messaging systems, allowing for flexibility and easy integration with other services or systems.
- **Focused on single responsibility:** Each microservice is designed to do one thing well, ensuring that its functionality is focused, easy to understand, and maintainable.
- **Resilience and fault tolerance:** Microservices architecture aims to prevent system-wide failures by isolating faults to individual services, allowing the rest of the system to continue functioning.
- **Discoverable:** Microservices use service discovery mechanisms to locate and communicate with other services in the system, enabling dynamic and flexible communication patterns.


© 2023 Akamai |

# Why You Should Consider Microservices

- **Faster time-to-market:** Microservices enable parallel development and deployment of individual components, accelerating the overall development process and reducing the time it takes to deliver new features.
- **Improved scalability:** Microservices can be scaled independently, allowing businesses to allocate resources more efficiently and handle varying workloads or traffic patterns more effectively.
- **Enhanced resilience:** The decentralized nature of microservices reduces the risk of system-wide failures, ensuring continuous service availability and better overall system reliability.
- **Flexibility and adaptability:** Microservices allow businesses to leverage diverse technologies and frameworks for different components, making it easier to adapt to changing requirements or incorporate new technologies.
- **Easier maintenance and updates:** The modular design of microservices simplifies system maintenance and updates, as individual components can be upgraded or replaced without affecting the entire system.


© 2023 Akamai |

# Microservices Best Practices

- **Keep services small and focused:** Microservices should be small, focused, and responsible for a single business capability. This makes it easier to maintain and update services and reduces the impact of changes on other parts of the application.
- **Design for failure:** Microservices should be designed to handle failure gracefully. This means implementing fault-tolerant and resilient architectures, such as retry mechanisms, circuit breakers, and bulkheads.
- **Use domain-driven design:** Microservices should be designed based on domain-driven design principles. This means modeling services based on business capabilities and using a common language to ensure that services are aligned with business needs.
- **Implement API gateways:** API gateways provide a central entry point for microservices and can help to simplify the communication between services. They can also provide security and rate limiting features.


© 2023 Akamai |

# Microservices Best Practices

- **Automate testing and deployment:** Microservices should be tested and deployed using automation tools such as continuous integration and continuous deployment (CI/CD) pipelines. This reduces the risk of errors and ensures that services are deployed quickly and consistently.
- **Use containerization:** Containerization provides a lightweight and portable way to package and deploy microservices. Using containerization can help to simplify the deployment process and improve the scalability of the application.
- **Monitor and log:** Microservices should be monitored and logged to ensure that they are performing as expected and to identify any issues or errors. This can be done using tools such as log aggregators and application performance monitoring (APM) tools.
- **Secure services:** Microservices should be secured using best practices such as authentication, authorization, and encryption. This ensures that the application is secure and that sensitive data is protected.


© 2023 Akamai |

# Containers

# What are Containers?

- **Lightweight virtualization:** Containers provide an isolated environment for running applications, sharing the host OS kernel but isolating processes, file systems, and network resources.
- **Portable and consistent:** Containers package applications and their dependencies together, ensuring that they run consistently across different environments, from development to production.
- **Resource-efficient:** Containers consume fewer resources compared to virtual machines, as they share the host OS kernel and do not require a separate OS instance for each container.
- **Fast start-up and deployment:** Containers start up quickly, as they do not need to boot a full OS, making them ideal for rapid deployment, scaling, and recovery scenarios.
- **Immutable infrastructure:** Containers are designed to be immutable, meaning they do not change once built, which simplifies deployment, versioning, and rollback processes, and helps ensure consistent behavior across environments.


© 2023 Akamai |

# Why You Should Consider Containers

- **Improved deployment consistency:** Containers package applications and their dependencies together, ensuring consistent behavior across different environments, simplifying deployment, and reducing the risk of configuration-related issues.
- **Enhanced scalability:** Containers enable rapid scaling of applications by quickly spinning up new instances to handle increased demand, optimizing resource usage, and improving overall system performance.
- **Cost-effective resource utilization:** Containers consume fewer resources than traditional virtual machines, allowing businesses to run more instances on the same hardware, leading to cost savings on cloud infrastructure.
- **Faster development and testing cycles:** Containers facilitate a seamless transition between development, testing, and production environments, streamlining the development process and speeding up the release of new features and bug fixes.
- **Simplified application management:** Container orchestration platforms like Kubernetes or Docker Swarm manage the deployment, scaling, and maintenance of containerized applications, automating many operational tasks and reducing the burden on IT teams.


© 2023 Akamai |

# Container Best Practices

- **Use lightweight base images:** Start with a lightweight base image, such as Alpine Linux or BusyBox, to reduce the overall size of the container and minimize the attack surface.
- **Limit container privileges:** Limit the privileges of containers to only those that are necessary for their intended purpose. This reduces the risk of exploitation if a container is compromised.
- **Implement resource constraints:** Set resource constraints such as CPU and memory limits to prevent containers from using too many resources and affecting the overall performance of the system.
- **Keep containers up-to-date:** Keep container images up-to-date with the latest security patches and updates to minimize the risk of vulnerabilities.
- **Use container orchestration:** Use container orchestration tools such as Kubernetes, Docker Swarm, or Apache Mesos to manage and scale containers across multiple hosts.


© 2023 Akamai |

# Container Best Practices

- **Use container registries:** Use container registries such as Docker Hub or Google Container Registry to store and manage container images. This makes it easier to share and deploy container images across multiple hosts.
- **Test containers thoroughly:** Test containers thoroughly before deploying them to production to ensure that they work as expected and are free of vulnerabilities.
- **Implement container backup and recovery:** Implement a backup and recovery strategy for containers to ensure that data and configurations can be easily recovered in case of a failure or disaster.


© 2023 Akamai |

# Serverless Computing

# What is Serverless Computing?

- **Event-driven architecture:** Serverless computing is based on an event-driven architecture where functions or code snippets are triggered in response to specific events or requests, rather than running continuously.
- **No server management:** In a serverless environment, the cloud provider handles the underlying infrastructure, including server provisioning, scaling, and maintenance, allowing developers to focus on writing code.
- **Automatic scaling:** Serverless computing automatically scales resources based on demand, ensuring that the required computing power is available when needed, without manual intervention.
- **Pay-per-execution pricing:** With serverless computing, businesses pay only for the actual compute time used during function execution, rather than paying for pre-allocated resources, leading to cost savings.
- **Enhanced agility and faster development:** Serverless computing enables developers to build and deploy applications quickly, without worrying about infrastructure management, leading to faster innovation and reduced time-to-market.


© 2023 Akamai |

# Why You Should Consider Serverless Computing

- **Cost efficiency:** Serverless computing pay-per-execution pricing model can lead to cost savings, as businesses only pay for the compute time they actually use, without needing to allocate resources in advance.
- **Improved scalability:** Serverless computing automatically scales resources to match demand, ensuring that applications can handle increased workloads without manual intervention or downtime.
- **Reduced operational overhead:** With serverless computing, the cloud provider manages the underlying infrastructure, freeing up IT teams to focus on application development, innovation, and other strategic initiatives.
- **Faster time-to-market:** The simplified development and deployment processes offered by serverless computing can help businesses accelerate the release of new features, updates, and bug fixes, enhancing their competitive advantage.
- **Flexibility and adaptability:** Serverless computing allows businesses to build and deploy applications using a variety of programming languages and technologies, making it easier to adapt to changing requirements or incorporate new technologies as needed.


© 2023 Akamai |

# Serverless Computing Best Practices

- **Use event-driven architecture:** Serverless computing is based on event-driven architecture, which means that functions are triggered by events such as HTTP requests, file uploads, or database updates. This can help to simplify the application architecture and improve scalability.
- **Use stateless functions:** Serverless functions should be stateless, meaning that they don't store any data or state between invocations. This ensures that functions are easily scalable and can be easily replaced if they fail.
- **Use short-lived functions:** Serverless functions should be short-lived, meaning that they should complete their work quickly and return a response. This ensures that resources are not wasted and that the function can scale quickly.
- **Monitor and log:** Serverless functions should be monitored and logged to ensure that they are performing as expected and to identify any issues or errors. This can be done using tools such as log aggregators and application performance monitoring (APM) tools.


© 2023 Akamai |

# Serverless Computing Best Practices

- **Secure functions:** Serverless functions should be secured using best practices such as authentication, authorization, and encryption. This ensures that the application is secure and that sensitive data is protected.
- **Test functions thoroughly:** Serverless functions should be tested thoroughly before deploying them to production to ensure that they work as expected and are free of vulnerabilities.
- **Use cost optimization techniques:** Serverless computing can be cost-effective, but it's important to use cost optimization techniques such as function optimization, resource sharing, and auto-scaling to reduce costs and improve efficiency.


© 2023 Akamai |

# Event-Driven Architecture (EDA)

# What is Event-Driven Architecture (EDA)?

- **Reactive to events:** Event-driven architecture (EDA) is a design paradigm where components within a system communicate and react to events or messages, rather than relying on direct, synchronous communication.
- **Loosely coupled components:** In an EDA, components are loosely coupled, enabling greater flexibility and resilience, as changes to one component do not directly impact others in the system.
- **Asynchronous communication:** EDA relies on asynchronous communication, which allows components to operate independently, improving system responsiveness and performance under variable workloads.
- **Scalability and resilience:** Event-driven systems can easily scale by adding or removing components as needed, and they are resilient to failures, as the system can continue functioning even if one component is unavailable.
- **Supports real-time processing:** EDA is well-suited for real-time processing and handling large volumes of data, as components can react to events and process data as it arrives, without waiting for a complete dataset.


© 2023 Akamai |

# Why You Should Consider Event-Driven Architecture (EDA)

- **Enhanced system flexibility:** The loosely coupled nature of event-driven architecture (EDA) allows businesses to easily modify, add, or remove components without affecting the entire system, making it adaptable to changing requirements.
- **Improved scalability:** EDA supports easy horizontal scaling, allowing businesses to handle increased workloads or traffic by adding more instances of components or services as needed.
- **Increased system resiliency:** EDA's asynchronous communication and decoupled components contribute to improved fault tolerance, as the failure of one component does not necessarily cause a system-wide outage.
- **Real-time processing capabilities:** EDA enables real-time processing of large data volumes and complex event patterns, making it suitable for businesses that require immediate insights or responses to rapidly changing conditions.
- **Optimized resource usage:** By reacting to events only when they occur, EDA helps optimize resource utilization and reduces the need for continuously running processes, potentially leading to cost savings and improved efficiency.


© 2023 Akamai |

# Event-Driven Architecture Best Practices

- **Use domain events:** EDA is based on domain events, which represent significant changes or occurrences in the system. Domain events should be defined in the domain language and should be based on real-world events.
- **Decoupling of components:** EDA emphasizes decoupling of components, meaning that each component should be responsible for a specific domain or business capability, and should communicate with other components through events. This helps to simplify the architecture and improve scalability.
- **Asynchronous processing:** EDA components should be designed to handle events asynchronously, meaning that they can process events in parallel and without blocking the main thread.
- **Idempotency:** EDA components should be designed to be idempotent, meaning that they can handle duplicate events without causing unexpected behavior or errors.


© 2023 Akamai |

# Event-Driven Architecture Best Practices

- **Event versioning:** EDA components should be designed to handle event versioning, meaning that changes to events can be made without breaking existing components.
- **Resilience:** EDA components should be designed to be resilient, meaning that they can handle failures and recover gracefully. This can be done using techniques such as retries, circuit breakers, and graceful degradation.
- **Monitoring and logging:** EDA components should be monitored and logged to ensure that they are performing as expected and to identify any issues or errors. This can be done using tools such as log aggregators and application performance monitoring (APM) tools.


© 2023 Akamai |

# Benefits of Portable Architecture

# Benefits of Portable Architecture

- **Improved scalability:** A cloud-native architecture enables applications to be more easily scaled up or down as demand changes, ensuring that they can handle peak loads without downtime or performance degradation.
- **Increased availability:** A portable architecture ensures that applications can be easily deployed across multiple cloud platforms, reducing the risk of downtime due to platform-specific issues.
- **Improved agility:** A cloud-native architecture enables developers to rapidly iterate and deploy new features and functionality, reducing time-to-market and improving competitiveness.
- **Reduced costs:** By adopting a portable architecture, organizations can take advantage of the pricing and feature differences between different cloud providers, reducing costs and improving cost-effectiveness.
- **Improved security:** A portable architecture enables applications to be more easily secured, with security features such as encryption and identity management integrated into the architecture itself.


© 2023 Akamai |

# Benefits of Portable Architecture

- **Increased resilience:** A portable architecture ensures that applications can survive hardware and software failures without interruption or data loss.
- **Easier management:** A portable architecture enables applications to be managed more easily, with tools for monitoring, automation, and orchestration that work across multiple cloud platforms.
- **Increased innovation:** A portable architecture enables organizations to take advantage of new and emerging technologies, such as AI and machine learning, to create innovative new applications and services.


© 2023 Akamai |

# Standardization

# Standardization

- **Use standard APIs:** Standard APIs should be used to ensure compatibility between the application and other systems. RESTful APIs are a popular choice for SaaS applications, as they are widely supported and easy to use.
- **Design for portability:** The application should be designed to be portable, meaning that it can run on any cloud platform. This can be achieved by using cloud-native technologies such as containers and Kubernetes, which provide a standardized way of deploying and managing applications across different cloud environments.
- **Use a modular architecture:** A modular architecture should be used to break down the application into smaller, more manageable components. This makes it easier to add or remove functionality as needed, and makes the application more flexible and scalable.
- **Implement automation:** Automation should be used to simplify and streamline the deployment and management of the application. This can be done using tools such as continuous integration/continuous deployment (CI/CD) pipelines and infrastructure-as-code (IaC) tools.


© 2023 Akamai |

# Standardization

- **Implement security:** Security should be implemented at all levels of the application, including authentication, authorization, and data encryption. This helps to protect the application and its users from attacks and breaches.
- **Use open standards:** Open standards should be used wherever possible to ensure interoperability and compatibility with other systems. This includes standards for data formats, protocols, and interfaces.


© 2023 Akamai |

# Resources

- **On-Demand Webinars**
  - Building SaaS Apps for Portability and Scale
  - The Move to the Distributed Cloud: Finding the Right Cloud for the Right Workload
- **Docs & Guides:**
  - Building SaaS Apps for Portability and Scale
  - How to Deploy Microservices with Docker
  - Deploying NGINX Ingress on Linode Kubernetes Engine
- **More Resources**
  - Is Your Cloud Development Strategy All Wrong?
  - Break Down Your Code: An Introduction to Serverless Functions and FaaS
  - Manage Serverless Kubernetes Applications with Knative
  - Future-proofing Success in the Cloud: The Power of Portability (recorded at Cloud Expo)
  - Portability and Interoperability: The Secret to Multicloud


© 2023 Akamai |

Are you looking for expert advice on cloud solutions? Schedule a free, one-on-one consultation with our Solutions Engineering Team to discuss your infrastructure needs and any questions about implementing microservices, containers, serverless computing, and more. Schedule some time, and our team will arrange a free brainstorming session.

**Schedule Now**


© 2023 Akamai |