Microsoft Cloud Workshop

Microservices architecture

Whiteboard design session student guide

December 2017

Contents

Microservices architecture whiteboard design session student guide

Abstract and learning objectives

This whiteboard design session is designed to help attendees gain a better understanding of implementing architectures leveraging aspects from microservices and serverless architectures, by helping an online concert ticket vendor survive the first 5 minutes of crushing load. They will handle the client's scaling needs through microservices built on top of Service Fabric, and apply smooth updates or roll back failing updates. Finally, students will design an implementation of load testing to optimize the architecture for handling spikes in traffic.

Attendees will learn how to:

- Implement scale and resiliency with Service Fabric
- Enable serverless solutions with Azure Functions
- Control API access with API Management
- Provide query flexibility with Cosmos DB

Step 1: Review the customer case study

Outcome

Analyze your customer's needs.

Facilitator/subject matter expert (SME) presentation of customer case study

Timeframe: 15 minutes

Directions: With all participants in the session, the facilitator/SME presents an overview of the customer case study along with technical tips.

1. Meet your table participants and trainer.
2. Read all of the directions for Steps 1–3 in the Student guide.
3. As a table team, review the following customer case study.

Customer situation

Contoso Events is an online service provider for concerts, sporting and other large event ticket sales. The current Contoso Events solution consists of a collection of web sites implemented in ASP.NET with a SQL Server back-end. The solution is currently hosted by Azure on Virtual Machines (VMs).

Contoso Events has experienced consistent growth trends and now has almost 1 million customers. They intend to further grow market share and increase sales by redesigning their current web sites to improve usability and conversion rates, improve responsiveness across devices. In addition, they will create mobile apps for iPhone, Android and Windows Phone devices. Following this, Contoso Events plans to extend its reach through partners by exposing its core event ticket sales and reporting APIs to partners. The plan is to retire and replace the existing solution to serve customers with a better experience – preserving code where possible, but migrating to a decoupled design with improved business agility.

To meet demand during peak periods they rely on a combination of auto-scaling and manual intervention to scale in advance of expected bursts of traffic – such as when a popular event has tickets first go on sale. The CIO, Steve Dormer, is concerned about system performance, scalability and associated costs. The company has already been experiencing more frequent, peak traffic periods and this has increased hosting costs. He would like to investigate ways to control exponential increases in hosting and related operational costs as they grow.

The company also has challenges rolling out new features and supporting new events on demand. Features have evolved in situ to where the solution has many interdependencies that increase the risk of regressions across features when changes are introduced. This is particularly challenging with the ticket ordering process as the data model for new events is often slightly different – which means that supporting new events may have impact on the user experience and UI, the middle tier and storage. Rolling out changes that impact this area while upwards of 50,000 users are actively placing orders, has proven to be a fragile process and requires them to schedule down time to ensure safe deployment.

The CIO has heard about microservices and serverless architectures, and is interested in exploring how Service Fabric and Azure Functions may help the team to be more agile in their development cycle with impact across features, support their goals for continuous delivery, help them with a lean and effective DevOps strategy, and ultimately help with cost control as they grow and support peak periods. The solution must be capable of supporting increased ongoing and peak loads yet also control costs by making better use of available infrastructure. The team also needs tools for monitoring health and more easily managing deployments, rollbacks and recovery in the event of failure.

In addition, the CIO is looking for a solid strategy around securely exposing and managing solution APIs for both internal and partner consumption. For partners, he is looking for a way to support a partner ecosystem that is easy to setup and manage.

According to the CIO, the current system topology handles the following core use cases:

- **Admin site**: Internal staff use a web application to manage events, customers, users and orders; and for executives to request reports
- **Consumer site (Web site)**: Customers search events, place ticket orders, return to view their orders and history of events
  - **Search events**: Customers will search for events they are interested in, from the event catalog available via the web site. The event catalog is managed by an internal administrative site
  - **Event details**: Customers may visit a specific event page to order tickets, from search results or a direct link from an email campaign or referral
  - **Ticket orders**: Customers select tickets to order from the event details page and are taken to an order page to complete the ticket order

        - This path receives the most traffic as customers compete to get tickets to a popular event that has just gone on sale
        - This is currently the focus of peak load and scale concerns
        - The CIO believes they should implement an asynchronous ordering process with email notifications - a pattern many busy ecommerce sites follow whereby the credit card is validated but the actual order is not confirmed until processed from a queue
    - **Payment**: Credit card payments are handled via a third-party payment processing service
- **Storage**: The event catalog, customer accounts, users and orders are currently stored in a relational SQL Server database
    - The CIO feels that a solution that reduces the overhead of state management and keeps the data closer to the compute functionality for faster retrieval will help them with future scale

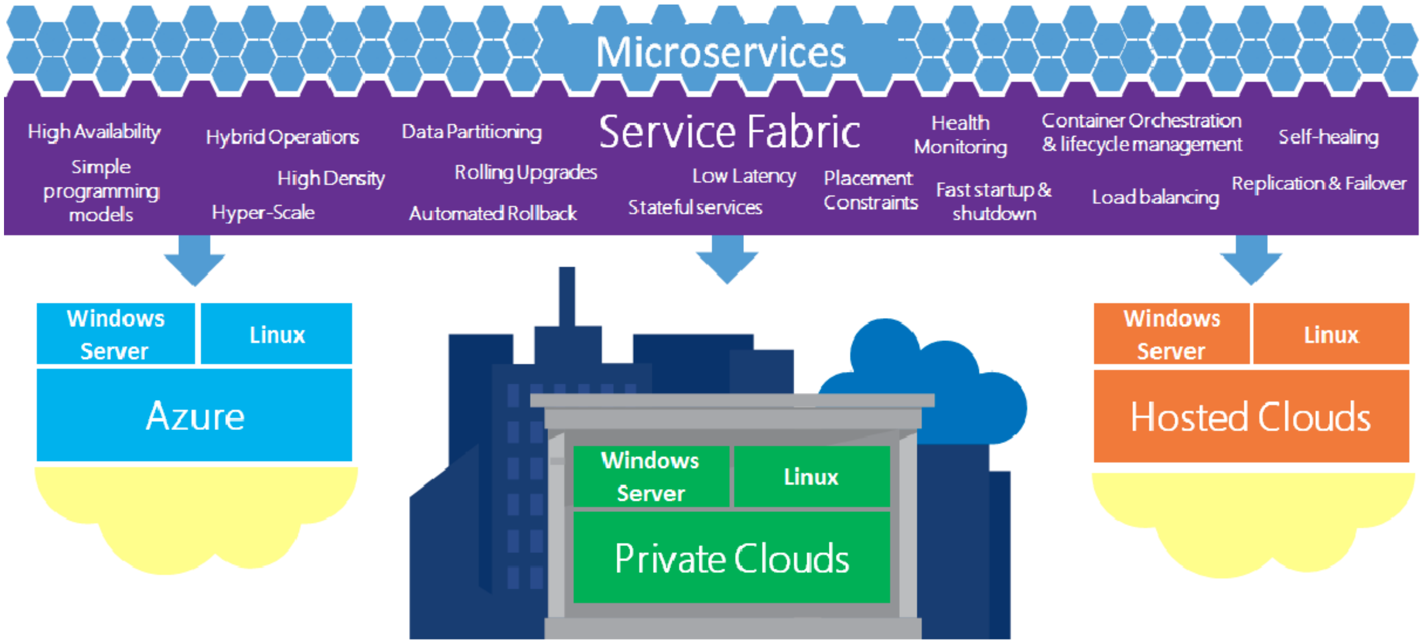
Customer needs

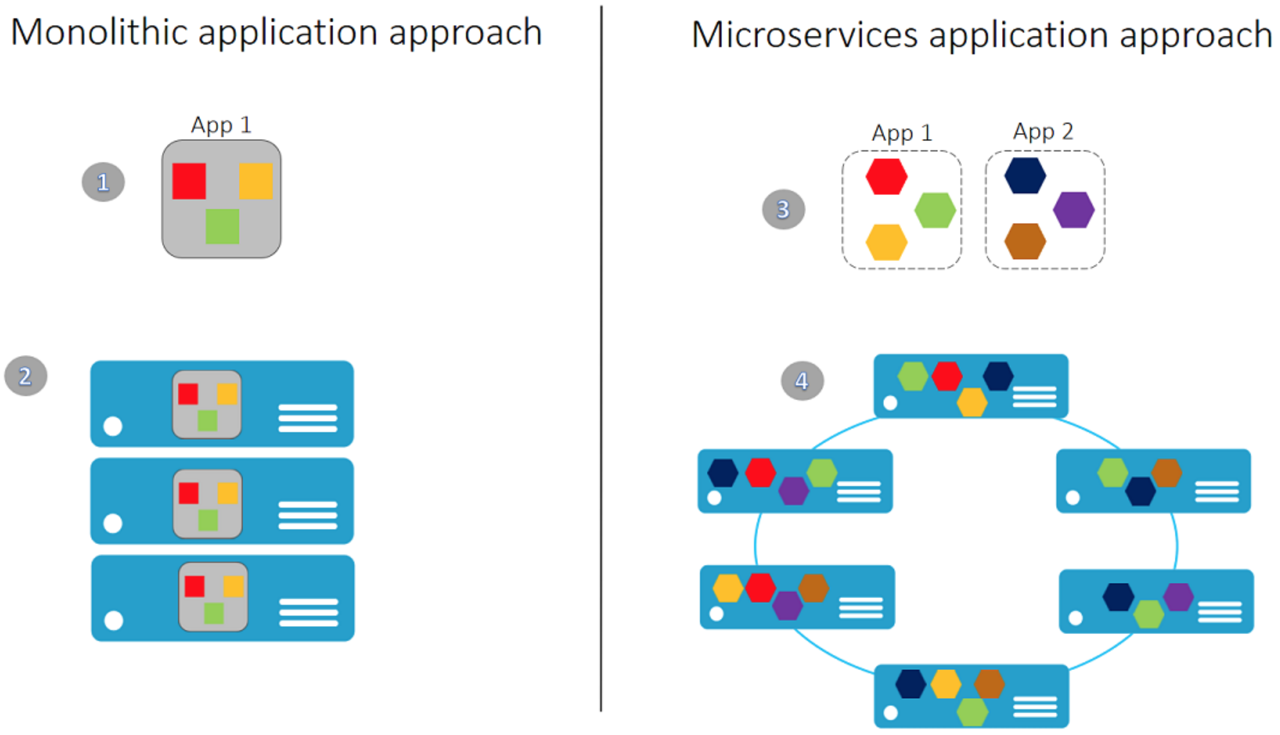
1. Event tickets can be ordered from multiple channels: the web site, new mobile applications, and third-party site and applications via available APIs.
2. Customers must be registered / logged in to place orders, so that they can login and find their orders, and for reporting and analytics purposes.
3. Internal staff will manage orders and view reports from the Admin site.
4. The ability to rapidly release new features that may involve UI, business logic and data model changes by reducing dependency across features.
5. Reduced overall downtime caused by system updates. Rollouts must be possible without scheduled downtime. Rollbacks must be possible in the event of failure.
6. The solution must be able to handle increased system load for ticket purchasing, including higher peak periods without excessive increases in management overhead and cost.
7. Operations management overhead must be improved through better system monitoring, visibility, self-healing services and auto-scale features.
8. The customer has decided to migrate from SQL Server to Cosmos DB for a more flexible schema and increased scalability across features.
9. A solution is required for securing and managing APIs used internally and by external partners, with the ability to easily publish APIs, version APIs, onboard consumers, control policy, monitor and audit usage.
10. The solution currently processes credit cards with a third-party payment-processing provider. This aspect of the solution will remain the same and requires integration into the new design.

Customer objections

1. While we are interested in the microservices approach, we are still comparing Service Fabric with PaaS features such as App Services and SQL DB. Service Fabric seems relatively new, while App Services and SQL DB have been around for some time.
2. Microservices architectures are completely new to the Contoso Events team. If we were to go forward with Service Fabric, we would like to understand what skills the team can carry forward, and how much of a learning curve exists.
3. We would like to understand if stateful services or stateful actors will help us with ticket ordering throughput, workflow and state management, and easier rollouts of changes to this process.
4. We are not clear how and where to incorporate stateful services and actors alongside other storage such as Cosmos DB. We need the ability to support robust ad-hoc queries against our system data such as events, customers, orders and related metrics – but would like to take advantage of the performance and reliability of Service Fabric stateful options as well.
5. Could we consider Azure Functions as an alternative back end implementation for our APIs?
6. We would like to understand more about the benefits of serverless architectures—in Azure does it mean only using Azure Functions or is there more to it?

Infographic for common scenarios



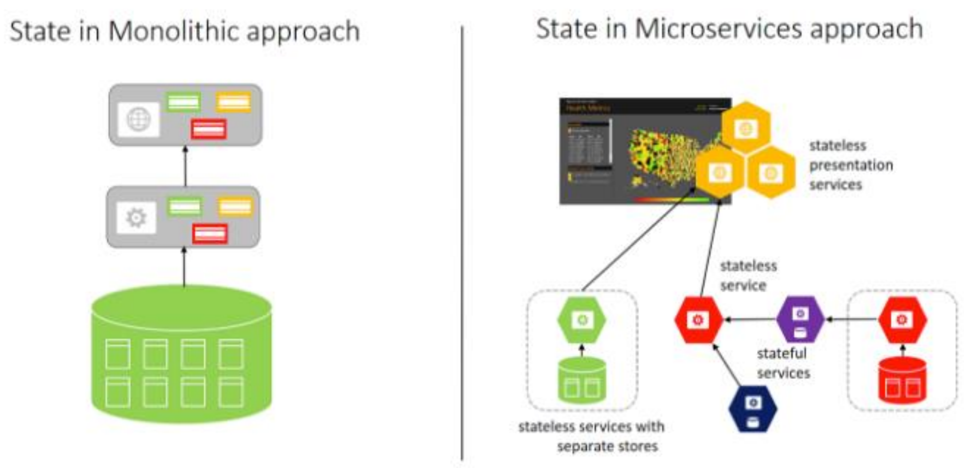

Monolithic application approach
App 1
1
2
Microservices application approach
App 1
App 2
3
4

Comparison between Monolithic and Microservices Approaches

A monolithic application contains domain-specific functionality and is normally divided by functional layers, such as web, business, and data (1). You scale a monolithic app by cloning it on multiple servers/virtual machines/containers (2).

A microservice application separates functionality into separate smaller services (3). The microservices approach scales out by deploying each service independently, creating instances of these services across servers/virtual machines/containers (4).

Reference: https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-overview-microservices


State in Monolithic approach
State in Microservices approach
stateless presentation services
stateless service
stateful services
stateless services with separate stores

Storage State between Monolithic and Microservices Approaches

The monolithic approach on the left has a single database and tiers of specific technologies. The microservices approach on the right has a graph of interconnected microservices where state is typically scoped to the microservice and various approaches are used to manage state.

Source: https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-overview-microservices

Step 2: Design a proof of concept solution

Outcome

Prepare to present a solution to the target customer audience in a 15-minute chalk-talk format.

Timeframe: 60 minutes

Business needs

Directions: With all participants at your table, answer the following questions and list the answers on a flip chart.

1. Who should you present this solution to? Who is your target customer audience? Who are the decision makers?
2. What customer business needs do you need to address with your solution?

Design

Directions: With all participants at your table, respond to the following questions on a flip chart.

High-level architecture

1. Without getting into the details (the following sections will address the particular details), diagram your initial vision for handling the top-level requirements for web sites, mobile applications, third party applications, access to APIs, compute and storage.
2. Based on the customer situation, what core services would you propose as part of the new microservices architecture? What state, if any, would those services hold? Illustrate with a diagram.

Scalability of ticket orders

1. Illustrate in more detail the Service Fabric services and components participating in a ticket order request.
2. Describe the scalability features of this design, including any partitioning strategies that are applicable.
3. Describe the resiliency of this use case. How can you create an asynchronous ticket order request and guarantee processing? Are there any potential points of failure? How will you address those?
4. Describe how you will enable external clients to reach stateless HTTP services exposed from the Azure load balancer.

Improving DevOps workflows

1. How would you structure the Visual Studio solution so that developers can run, debug, and publish the entire solution but also be able to publish and upgrade individual microservices (could be one or more service grouped together)?
2. Describe to the customer how they can upgrade services in situ and preserve state; handle rollback and roll forward; and service self-healing features.
3. Explain how the Service Fabric cluster handles auto-scaling. How does Service Fabric help the customer to make better utilization of their compute resources?
4. How would you recommend the customer plan for high availability (HA) in this solution?
5. Explain to the customer how Service Fabric can help the customer have visibility into overall solution health.
6. How can you update cluster settings after the fact? What kind of settings might you want to update?
7. How will you keep your cluster up to date with the latest Service Fabric SDK?

Controlling access to APIs

1. Describe how API Management may be useful to protect access to APIs exposed by the solution. How would you identify the user and the API consumer or application?
2. How will you protect Service Fabric APIs from outside callers exposed through API Management?

Prepare

Directions: With all participants at your table:

1. Identify any customer needs that are not addressed with the proposed solution.
2. Identify the benefits of your solution.
3. Determine how you will respond to the customer's objections.

Prepare a 15-minute chalk-talk style presentation to the customer.

Step 3: Present the solution

Outcome

Present a solution to the target customer audience in a 15-minute chalk-talk format.

Presentation

Timeframe: 30 minutes

Directions

1. Pair with another table.
2. One table is the Microsoft team and the other table is the customer.
3. The Microsoft team presents their proposed solution to the customer.
4. The customer makes one of the objections from the list of objections.
5. The Microsoft team responds to the objection.
6. The customer team gives feedback to the Microsoft team.
7. Tables switch roles and repeat Steps 2–6.

Wrap-up

Timeframe: 15 minutes

- Tables reconvene with the larger group to hear a SME share the preferred solution for the case study.

Additional references

Item	Description	Links
Service Fabric	Overview of load balancing and addressing services	https://azure.microsoft.com/en-us/documentation/articles/service-fabric-connect-and-communicate-with-services/
Service Fabric	HA Configuration (demo only)	https://alexandrebrisebois.wordpress.com/2016/05/31/deploy-a-geo-ha-service-fabric-cluster-on-azure/
Azure Functions	Process events with serverless code	https://azure.microsoft.com/en-us/services/functions/
Cosmos DB	Guidance on core features and partitioning	https://azure.microsoft.com/en-us/documentation/articles/documentdb-introduction/
Azure AD	B2C	https://azure.microsoft.com/en-us/services/active-directory-b2c/