Application resilience made easy with Azure

Issued by Bluegrass Digital

27 Jun 2023

There are so many great examples of resilience in nature. Antarctica's emperor penguins travel many miles to reach their breeding ground and once the female has laid an egg, the male penguins huddle in large groups and endure months of freezing cold weather incubating these eggs in some of the worst conditions on earth. And during heavy rains and floods, fire ants cluster together in massive groups to create water-repellent lifeboats that can float for weeks on end without any of the ants drowning.

Within the application development space, resilience is all about designing to withstand failures; making it possible to react to big events or issues in one area while still providing the best possible service across others. It's not about avoiding failures but accepting that things will go wrong and finding ways to respond to incidents in a way that avoids downtime or data loss.

Because, let's face it, outages happen. In 2022, there were a number of high-profile system outages – from Twitter, Zoom and AWS to Google and WhatsApp – that left users unable to connect to cloud services, join meetings or search for information. These incidents bring the importance of system resilience and redundant architecture into sharp focus.

How is application resilience different?

With a traditional application, a request comes in, you do some handling, fetch some data and then respond to the user. But if any of these requests are even vaguely computationally expensive, or you add additional load to a request, you're going to have a problem. When the input size is too great, it requires a relatively large number of steps to complete and can cause your application to go down. This is a problem that affects everyone – regardless of business size or industry.

Luckily for modern businesses, there are a number of coding and app development practices that you can follow to mitigate failure and put the right recovery mechanisms in place. Some of these are unpacked below.

Leverage caching

Depending on the type of request coming in, it might be possible to serve a user with "stale" data. Obviously this is a more suitable strategy for some requests and not for others. And not everything is cacheable. If you run an e-commerce store, for example, the majority of your requests will be coming in for your list of product categories. This information is highly cacheable because your product categories aren't likely to change that frequently. But if you are handling the ticketing for a big concert, you can't afford to have your information be even slightly out-of-date. One of the big benefits of caching is that you can vary your cache depending on different factors – so data that is less variable can be cached for longer periods.

Scaling up or scaling out

Modern applications need to be built to scale. Within Azure App Service, there's this concept of scaling up resources and scaling out resources. The former will respond to additional load by spinning up something entirely new and moving all the traffic across to that virtual machine. This is a suitable strategy when an application can't scale out but it is time-consuming, expensive and often the application goes down in the process. When scaling out, the application will handle the load from a spike in traffic by distributing this load across different instances depending on the volume of requests coming in at any particular time. In this way, the load is balanced because requests are routed evenly to instances with capacity and the additional load is shared.

Consider availability zones

Azure availability zones protect applications and data from failures and enhance application resilience. Like the emperor

penguins snuggling up to keep warm or the fire ants banding together to form a raft, these zones act as a bit of a safety net when you hit tough times. All resources within Azure can be configured to work across different availability zones so that if there's a fire or a local outage in one part of the data centre, your app won't be affected.

Embrace serverless functions

If you have something that is extremely computationally expensive, you should consider using serverless functions. Rather than having an application that is running all the time, you can construct your APIs using serverless functions. These aren't running by default but as soon as a request comes it, it spins up an instance, handles your request and then spins this function back down again. This is a great way to architect an application without having to worry about the traditional scaling problems and is ideal for handling high volume, periodic load.

Need a partner to help you develop your next mobile application? At Bluegrass Digital we have specialist teams who have the skills to deliver a solution that meets your unique requirements and that is developed with resilience in mind. To find out more, get in touch, here.

- **Why Umbraco CMS continues to rise as an enterprise CMS leader** 26 Apr 2024
- **Welcome to the era of the AI co-pilot** 11 Apr 2024
- **5 dos and don'ts to secure customer loyalty** 12 Mar 2024
- **Bluegrass successfully transforms Kenya Airways website** 22 Nov 2023
- **A roadmap for marketing personalisation success** 16 Nov 2023

Bluegrass Digital

bluegrass digital

We help businesses transform and succeed in a digital world through insight-led customer experience, innovation and technology built to scale.

Profile | News | Contact | Twitter | Facebook | RSS Feed