Scalable GIS

John Maloney

The foundation of the cloud, and arguably it's greatest feature, is that it is fundamentally scalable. As traffic and data volumes increase so can the ability of applications that are made for the cloud. Geographic Information Systems are in the early stages of understanding and using the tools and scalability implicity supplied by cloud infrastructures, this allows a lot of room for growth. In the application defined in this article two typical scenarios in a GIS are used to be a model for defining a scalable architecture, tile rendering and geospatial analysis. These tools are not actually implemented, but are used to define the pathway to implementation using a pattern that inherently creates scalability. This pattern is known as the Publisher/Subscriber pattern and is described as:

"a messaging pattern where senders of messages, called publishers, do not program the messages to be sent directly to specific receivers, called subscribers, but instead categorize published messages into classes without knowledge of which subscribers, if any, there may be. Similarly, subscribers express interest in one or more classes and only receive messages that are of interest, without knowledge of which publishers, if any, there are." - Publish-subscribe Pattern, Wikipedia

By leveraging the Azure Service Bus, which is an implementation of the Publisher/Subscriber pattern, the application defined in this article can explore the scalability offered by the cloud. To create the ability to allow for rapid scaling the application must be produced in a way that allows an instance of the application to be brought into "existence" rapidly and in an automated way. The best tool for this scenario is containerized software. By using the tool Docker an application can be isolated to a subset of behavior and therefore is much easier to scale up and down as demand increases and decreases.

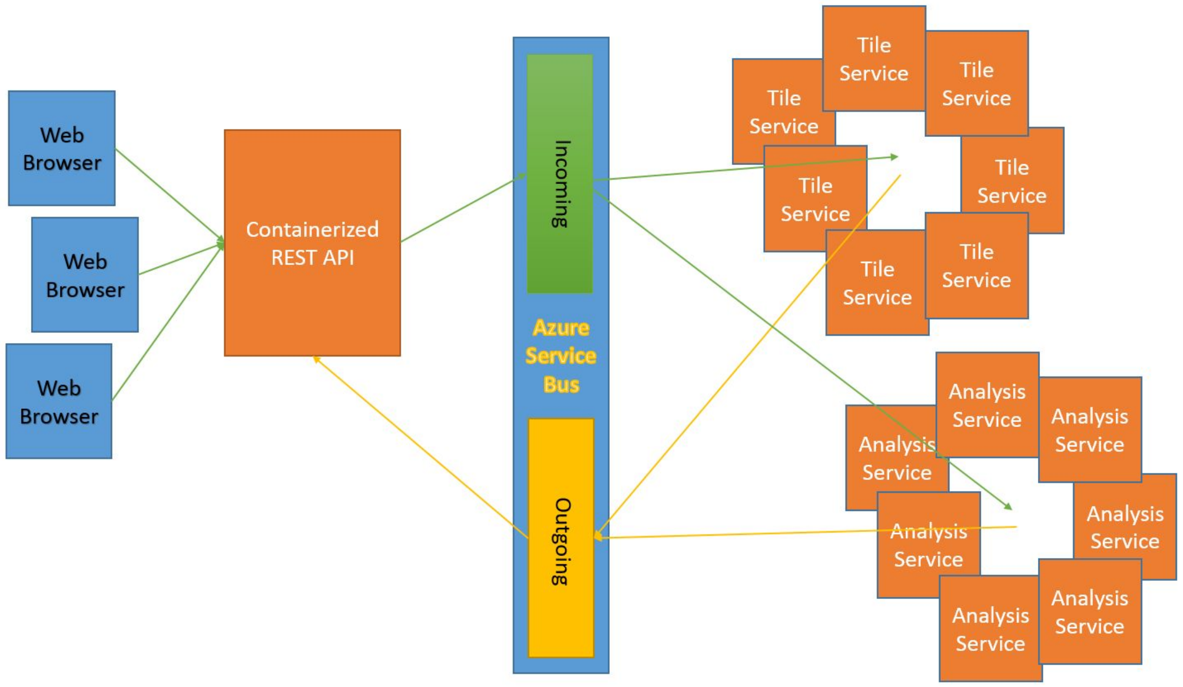

Figure 1 - Cloud based implementation of the Publisher/Subscriber pattern and Docker containers.

Using the Azure Service Bus and Docker containers the application architecture becomes disconnected from each layer (figure 1). This “disconnectedness” allows moving from the client web browser to the service bus and them out to all the containers that will implement the behavior required for producing geospatial tiles and analysis effectively. The service bus becomes the catalyst for scalability and allows sections of the application to become less dependent on other sections. This has the side benefit of allowing teams to own and deploy feature on their own schedule. Thresholds must be set so that as pressure increases on the application the container instances can be replicated to provide enough instances to respond to the load. As soon as the pressure decreases the instances can be shut down and even destroyed to save on expenditures. With this type of architecture the technologies used do not matter anymore, it becomes about the best tool to implement a geospatial procedure with the most efficiency.

With this cloud based architecture in place a GIS is allowed room to operate on large datasets and produce deep analysis of geospatial data while still providing for the more basic features of tile rendering and map styling. It seems that with the adoption of the cloud as the new field for application development that GIS applications can benefit greatly from the underlying scalable nature. This will require that new standards and avenues are explored and documented and that GIS providers give guidance on best practices and common pitfalls and misconceptions.

References

Publish-Subscriber Pattern () Wikipedia, Retrieved on December 12th 2017 from
https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern

Microsoft Service Bus Tutorial, Retrieved on December 12th 2017 from
https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-dotnet-get-started-with-queues

Docker Tutorial, Retrieved on December 12th 2017 from
https://docs.docker.com/engine/examples/dotnetcore/#create-a-dockerfile-for-an-aspnet-core-application

Github Repository for the application - https://github.com/johnmaloney/WebGIS