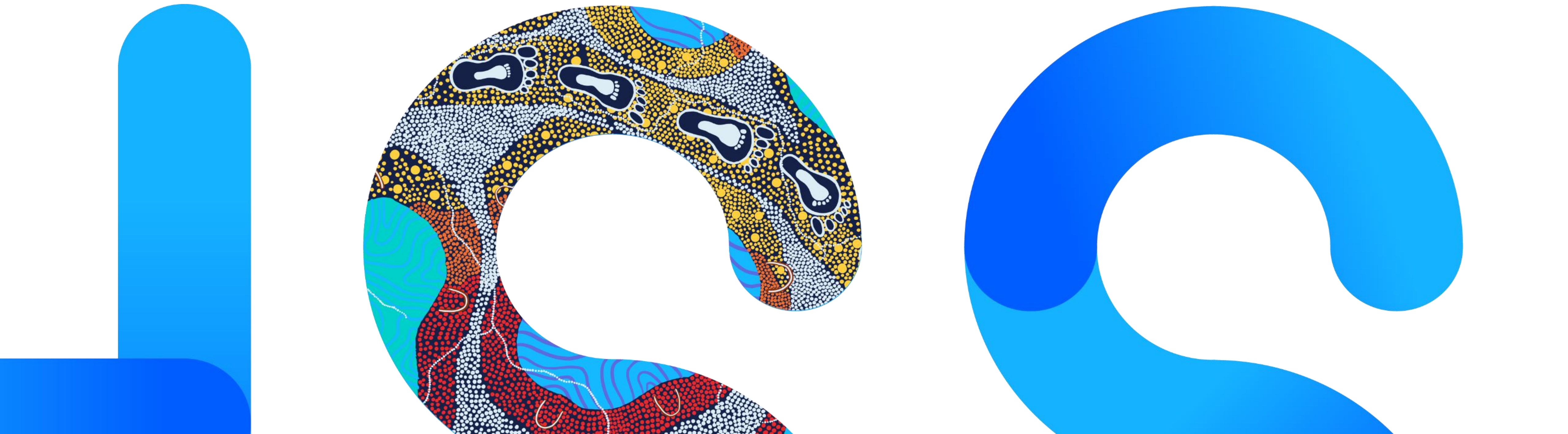
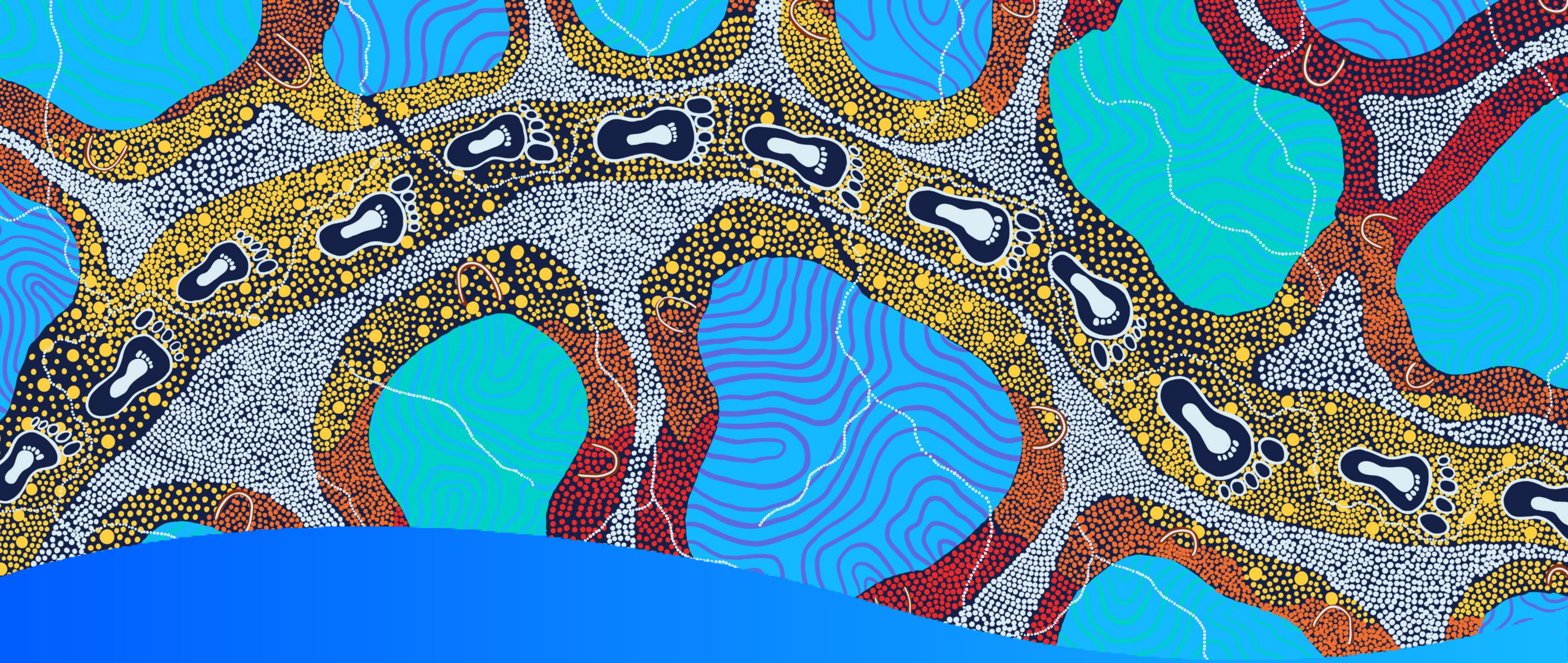
Day 4 – Azure Functions

Serverless function in Azure





HSS acknowledges the traditional custodians throughout Western Australia and their continuing connection to the land, waters and community. We pay our respects to all members of Aboriginal communities and their cultures, and acknowledge the wisdom of Elders both past and present.

Quick Recap of Day 3

1. Different application hosting options in Azure
2. Azure App Services
    1. For hosting web/HTTP-based workload
    2. Supported programming language - .NET, .NET Core, JAVA, Python, Ruby, NodeJS, PHP
    3. Scaling Feature – In/Out, Up/Down
    4. DevOps Optimised
    5. App Service Plan – the cost of the plan to host your app
    6. App Service Environment – dedicated, isolated environment to host your app
    7. Authentication integration
    8. Service Connector – allows your app service to connect to backend services (SQL, Storage, Service Bus etc.)
3. Homework solution

Serverless in Azure

What is serverless?

From Microsoft: Serverless computing enables developers to build applications faster by eliminating the need for them to manage infrastructure. With serverless applications, the cloud service provider automatically provisions, scales, and manages the infrastructure required to run the code.

The different type of serverless offering from Azure

1. Azure Functions – code-first development that reacts to an event
2. Azure Logic Apps – designer-based development that reacts to an event
3. Power Automate (prev. known as Microsoft Flow) – very similar to Azure Logic Apps but with less connector and more tailored to the Power Platform-type of development
4. Azure App Service WebJobs – Similar to Azure Functions but it can only run on App Service and can only be written in C#

What is Azure Functions?

Serverless solution that allows you to not worry about infrastructure hosting and write less code to perform a task.

It’s a “compute on-demand” type of “application” where it reacts to an event (**trigger**), process the event (**action**) and produce some outputs from the processing (**output**).

It only reacts to one event on every execution.

Azure Functions – features & concepts

1. Different Hosting Plan type
    1. Consumption plan – Pay as you go, scales as you go, not warmed instance that may result in delay start
    2. Premium plan – “Workers” are pre-warmed which means no delay start, ability to connect to virtual network, run on more powerful instance (providing more processing power)
    3. Dedicated plan (aka App Service Plan) – you just need to pay for the App Service Plan, predictive costing
2. Function timeout – Running functions will timeout to stop execution and your function must respond within the time limit
    1. Consumption plan – Default 5 mins, Maximum 10 mins
    2. Premium and Dedicated – Default 30 mins, Maximum unlimited
3. Trigger – event that the Azure Function reacts to
4. Action – perform something against that event
5. Output – outcome of the processing
6. Ability to connect to our on-premise workload using Virtual network – only available on Premium and Dedicated plan
7. Durable function – Extension of Azure Functions to allow you to write stateful functions.
    1. Azure functions by default is stateless ie. it reacts to something, process it and that’s it.
    2. Different durable function patterns: Durable Functions Overview - Azure | Microsoft Learn

Playtime!

1. Open your WhereIsPatient solution from last week.
2. Create a new project from the “Azure Functions” template.
    1. Functions Worker: .NET 6.0
    2. Function: Queue Trigger
    3. Use Azurite for runtime storage account – **Untick** this unless you have Azurite installed on your machine
    4. Authorisation Level: Anonymous
    5. Connect to dependency – Choose Azure Storage
        1. Choose your Azure Storage and proceed.
3. Once the project is created, install these nuget packages:
    1. **Microsoft.Extensions.DependencyInjection**
    2. **Microsoft.Azure.Functions.Extensions**
4. Add a reference to the database project.
5. Create a class file called “**Startup.cs**” and add the code that will be sent in Teams.
6. Make sure there is a queue called “**patient-message-queue**” in your Azure Storage.
7. Create the function to process the message and write to DB.