https://tinyurl.com/TechBashAzureTradeoff

Azure feature tradeoff analysis

JAMIE ROMANOWSKI

A FEW THINGS ABOUT ME

@JamieRomanowski

Green Leaf
CONSULTING GROUP

What this talk is

Review features in azure with overlapping capabilities

Learn how to use a tradeoff analysis

Discuss how to plan a Cinco De Mayo party

https://tinyurl.com/TechBashAzureTradeoff

What is a tradeoff analysis?

A tradeoff analysis is an approach for optimizing your decision making process for your specific needs based on comparing the impact of competing features and aligning them according to your needs.


TRADEOFF ANALYSIS
SO HOT RIGHT NOW
imgflip.com

Tradeoff Analysis is very relevant currently

- Less ecosystems are single tool focused, more freedom to diverge exists
- More **highly opinionated** and evolving products and technologies available to choose from
- Increase in distributed systems / microservices
  - expectation on loosely coupled services
  - All could be in different technologies
- Team empowerment leads to less centralized decisions and more diverse technologies and tools
- Easier to change and swap out technologies
- With the increase in use of open source technologies, its not just buy vs build – now it's buy, build, or leverage

< 5 minute non-technical tradeoff analysis example

tortilla with cheese, meat, and/or vegetables…

- ✓Soft or hard
- ✓Can vary the toppings or ingredients
- ✓Could do a taco bar!

- ✓Could eat cold
- ✓Typically done with flour tortillas instead of corn
- ✓Great when you need to eat something one handed or on the go.

- ✓Typically baked and served with a sauce
- ✓Great when you want a more elegant dinner

Where deeper analysis is important

- Your specific scenario
  - Tuesday night dinner
  - Hosting a party where people will show up at different times in the day
  - Dorm room with only a hot plate
  - A movie theater that wants to plan something for the Three Amigos remake
  - Tailgating before an event with or without a grill
  - A catered party at your house
  - Happy hour on Cinco de mayo …

Cinco De Mayo Happy Hour Tradeoff Example

- You invited a few friends over for a Cinco De Mayo party at your place.
- One of your friends can't eat cheese.
- One of your friends is gluten sensitive
- One of your friends is allergic to onions
- What is the right thing to do…

Cinco De Mayo Happy Hour Tradeoff Example

- Assumptions
  - Everyone is coming from work so they should show up around the same time
- What do I care about?
  - I want to show off just how good of a cook I am
  - Everyone can eat something
  - You want to have fun too and not have to spend a lot of time cooking during the party
- What do I not care about?
  - Cost (all are about the same)
  - Prep time (you are taking the afternoon off – flex time!)

Happy hour on Cinco De Mayo Tradeoff chart

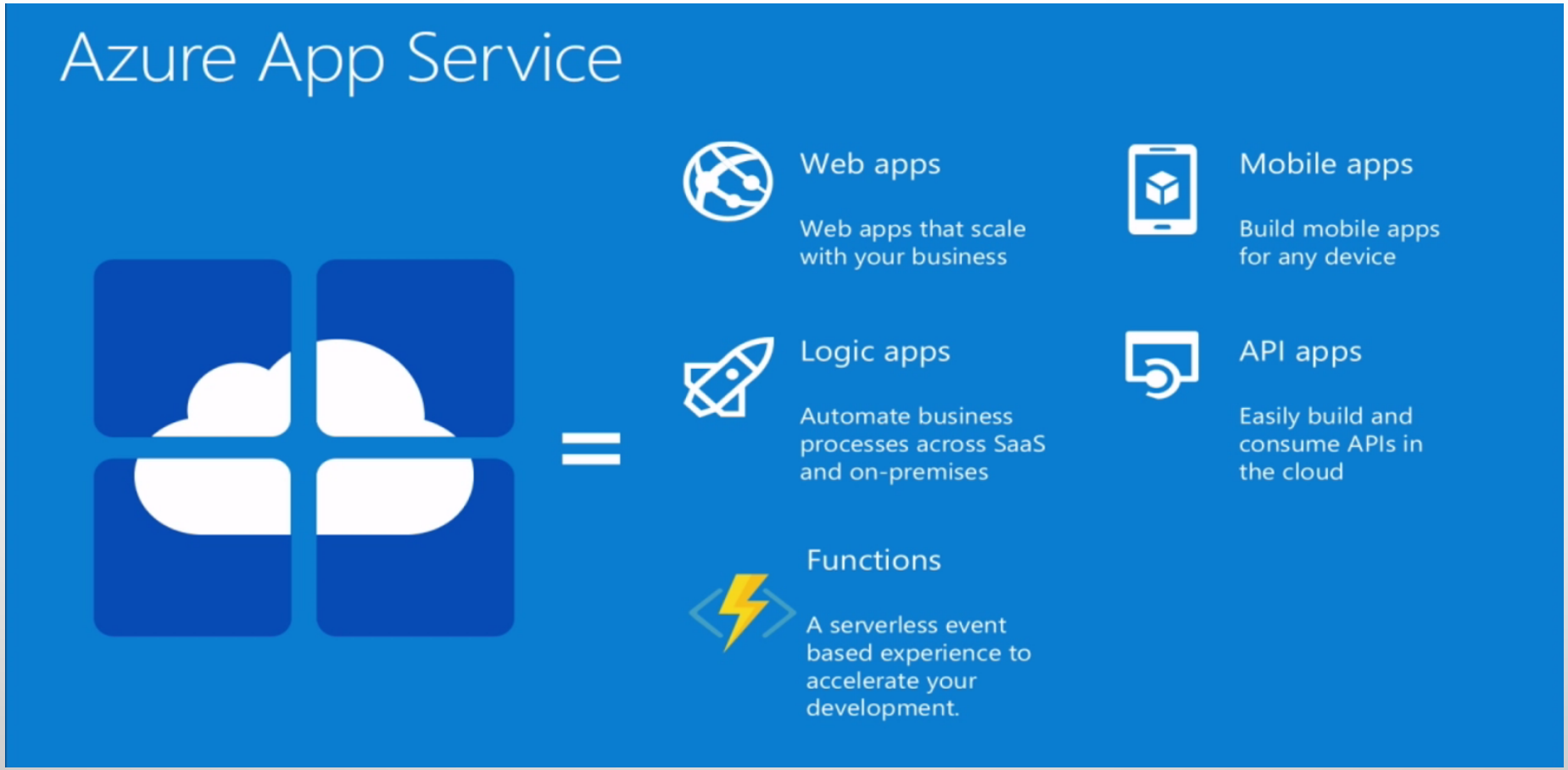


Show off my mad cooking skills	
No-cheese	
No gluten	
No onions	
Low maintenance during party	

Pro-tip

- Focus on the dependencies / links that's where the key investigation and research points are
- You don't have to do a weight if you don't need to – it is optional
  - Prioritize and only include what you care about
  - Often times you spend more time "discussing" the weighting of decisions when if you just put down the analysis there is an obvious choice waiting for you.
  - If you really have to do a weight consider using a how much do you care scale/categorization – "A lot", "Some", "Little" – be creative, emoji, # of spicy peppers, etc.

Azure Capabilities



- Externally hosted services that allow you to run your business functions (compute) and not have to buy, maintain, or protect (to a degree) your own servers, storage, etc.

Azure app services

Azure web jobs

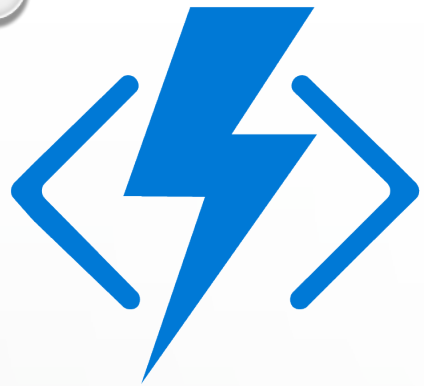
Azure functions

Azure App Service

- **Azure App** Service is an HTTP-based service for hosting **web applications**, REST APIs, and mobile back ends. You can develop in your favorite language, be it .NET, .NET Core, Java, Ruby, Node.js, PHP, or Python. **Applications** run and scale with ease on both Windows and Linux-based environments
- Hosted on managed VMs
- Whole system deployed together
- Pricing
  - billed at an hourly rate
  - service plans vary per size and scaling instances, VM power, memory, etc from $0.10 to $0.40 per hour

Azure Web jobs

- Feature of app services that enables you to run a program or script in the same context as the web app.
- Continuous vs Triggered
  - Continuous runs on all instances or a single one. can debug remotely
    - Always On feature - is always running (5 min gap max),
  - Triggered runs when triggered (timer or manual) – always runs on single instance (load balanced)
- There are no limitations on time (remember they run on your web servers…)
- For time triggers - execution can be at most 1 per minute
- Pricing is
  - Cost included in your app service



Azure Functions

- Azure functions are a premier service offering built directly on the WebJobs SDK
- Trigger events – timer, storage queues and blobs, service bus, cosmos db, event hub, http, http webhooks, azure event grid
- Languages – C#, F#, JavaScript, Java, Python, Poweshell (New)
- Can run and test locally
- Integration with other services like logic apps

Azure Functions – hosting plans

- You need a storage account to have an Azure function – this is billed separately.
- Consumption Plan
  - Pay only for what you use
  - Scale automatically
  - Limitation – 5 minute maximum*

Azure Function pricing

- # Executions
- Execution time in GB/Seconds
- Any storage you use

- Azure App Service Plan
  - Runs like an app service on the app service plan with no time limits
- Premium Plan ~~(Preview)~~ NEW
  - Consumption plan + enhanced performance and VNET access
  - Higher hourly rates
  - No slow starts
  - Unbounded run time limit --> default 30 mins

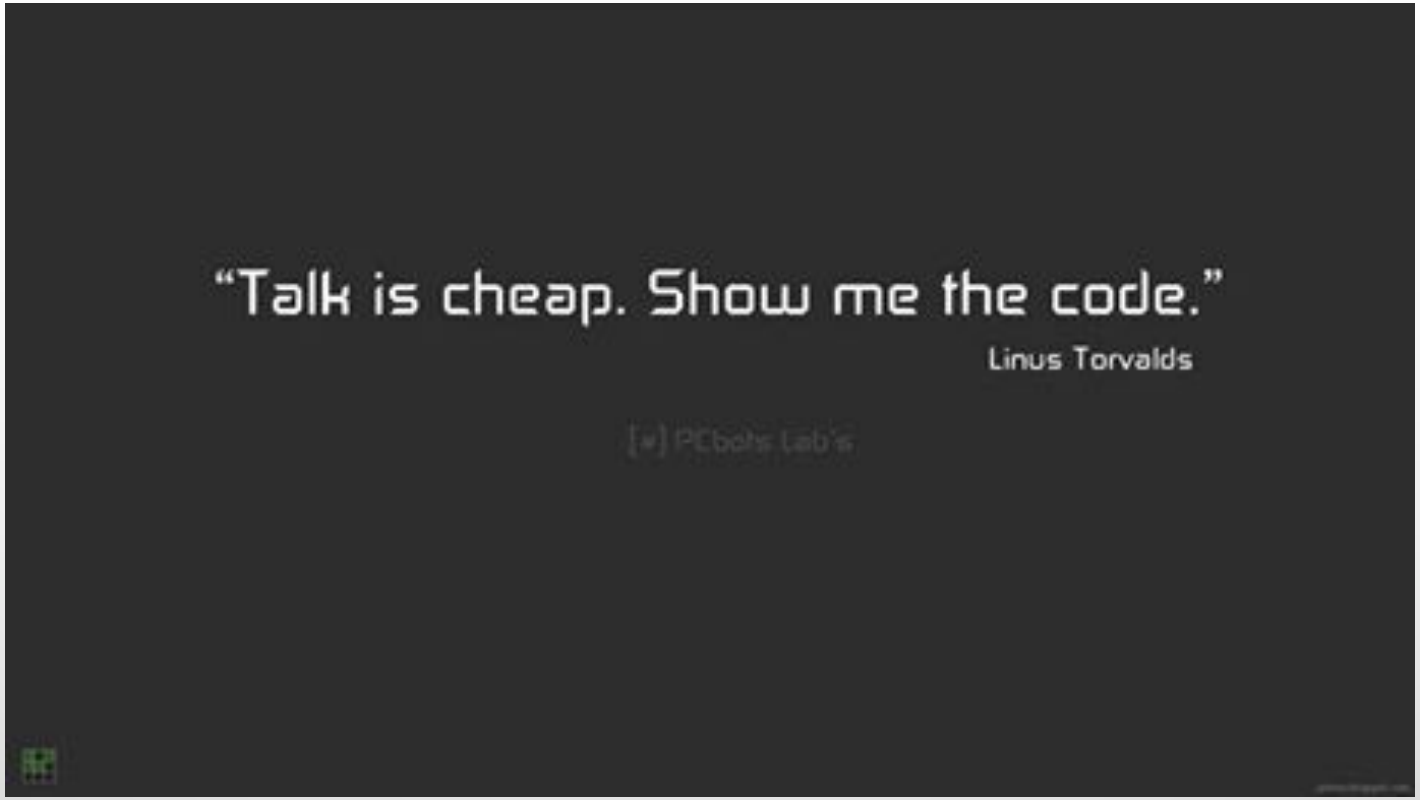
https://mikhail.io/serverless/coldstarts/azure/

Azure Functions vs Web Apps Scalability and Performance

- Other than slow starts there are not many other considerations points to consider across the products.
  - After ~20 minutes Azure will shut down your app and it will need to be reloaded
  - On average 2-3 seconds to reload
    - Depending on size, **dependencies**, OS, and deployment method it can take longer
  - Hack - add a timer based function to Keep Alive for ~ 2,976 calls a month

- In my experience, scaling problems can be unique and require unique solutions to keep them cost effective – so would need further analysis.

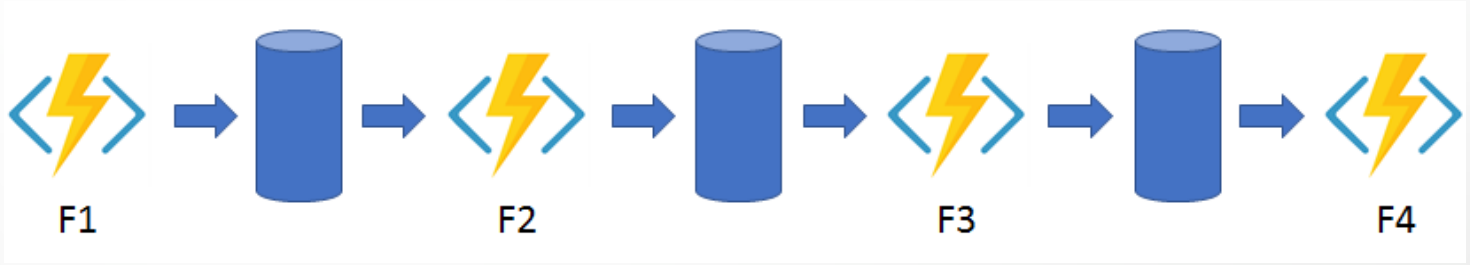
Azure Functions – Demo

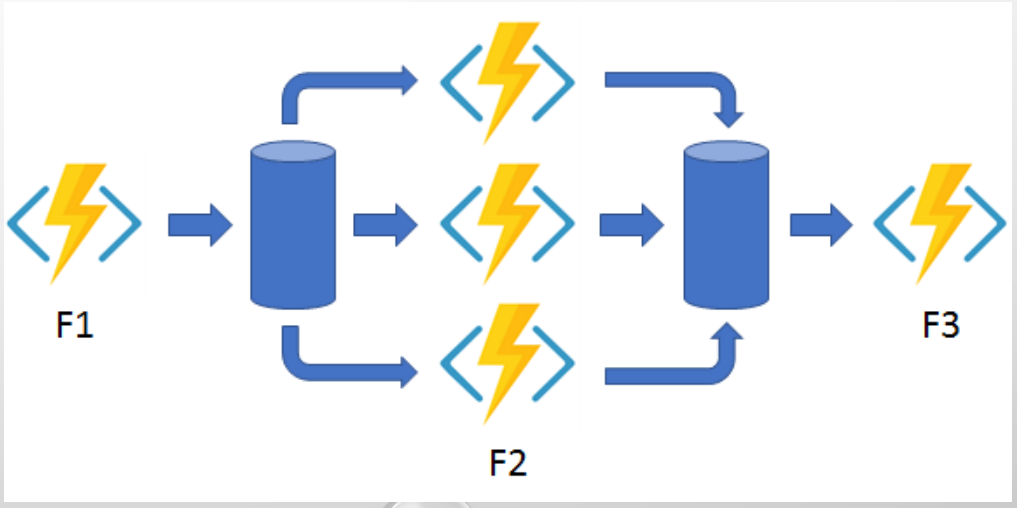


- DEMO – what does an Azure Function project look like…

Azure Functions – Durable

- Common Patterns
  - Function Chaining
  - Fan Out / Fan In (sagas)
  - Async HTTP calls
  - Polling / Timer / Monitoring
  - Waiting for human interaction





Azure Functions – hosting plans

- DEMO – a more advanced durable function

Azure Functions – before you jump

- What is really hard about functions / serverless / event based?
  - DEBUGGING!!!
  - Designing
  - Deconstructing

Azure Functions – Proxy

- Function proxy
  - A proxy is a function that is triggered by an HTTP request and allows you to manipulate the request and/or the response.
- Why would we use this
  - Advanced routing for scenarios like versioning (v-current, v-next, etc.)
  - Can help control versioning or mock out test scenarios
- Please remember
  - This is an additional cost, as in another function that will be called in addition to the API call and it will remain active above the time of the other call

Azure Functions Apps Jobs - What we might care about?

- Overall Cost
- Billing transparency
- Developer experience
- Operations experience
- Scalability & Performance
- Architecture and Deployment strategies

Azure Functions vs Web Apps - pricing

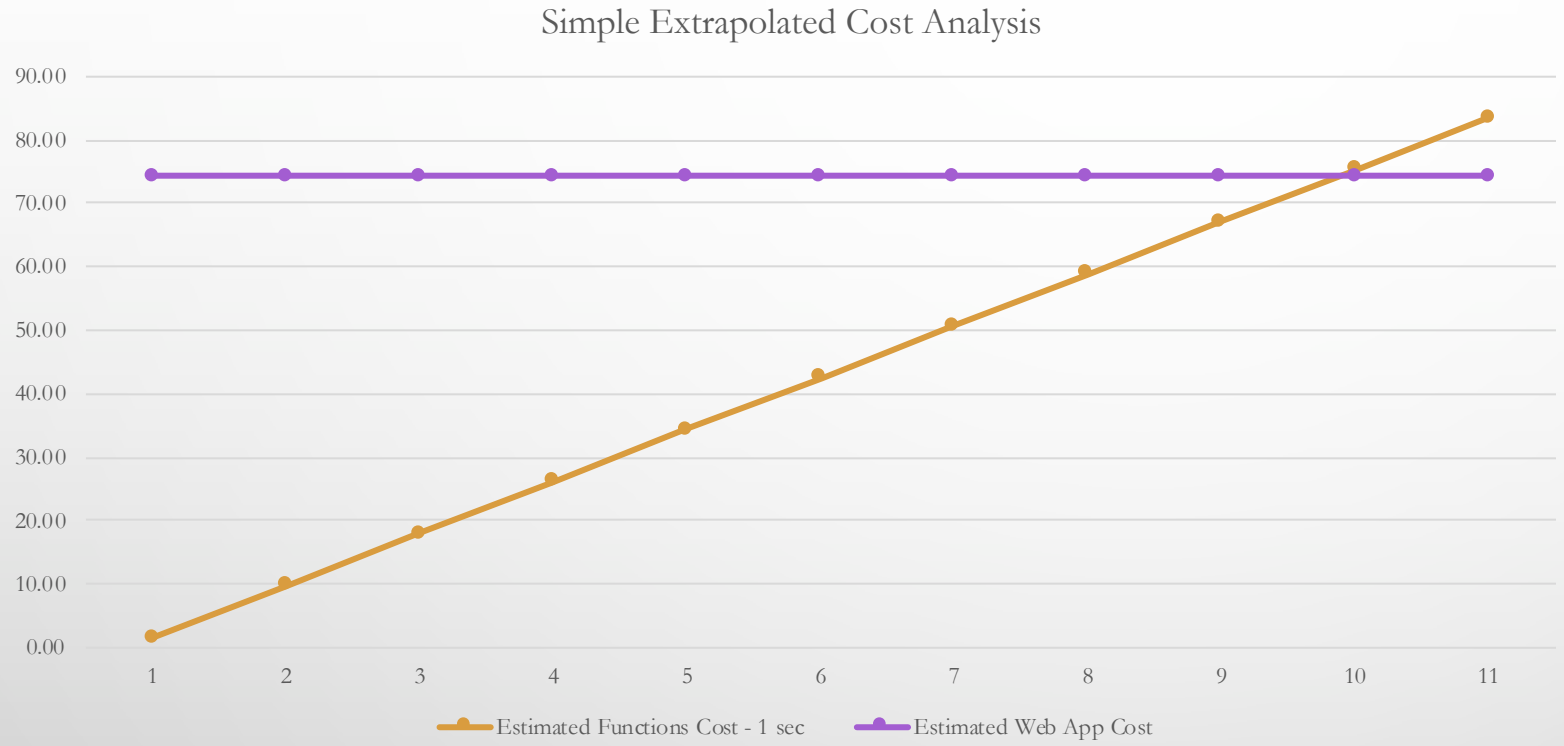
Azure App Service

- S1 → .10/hour 1 Core 1.75GB RAM ~$74.4 month
- S2 → .20/hour
- S3 → .40/hour
- P1 → .20/hour
- P2 → .40/hour
- P3 → .80/hour

Azure Function pricing – consumption plan

- # Execution * seconds (1M free)
- Execution memory use in GB/s (400K free)
- Any storage

Azure Functions vs Web Apps - pricing

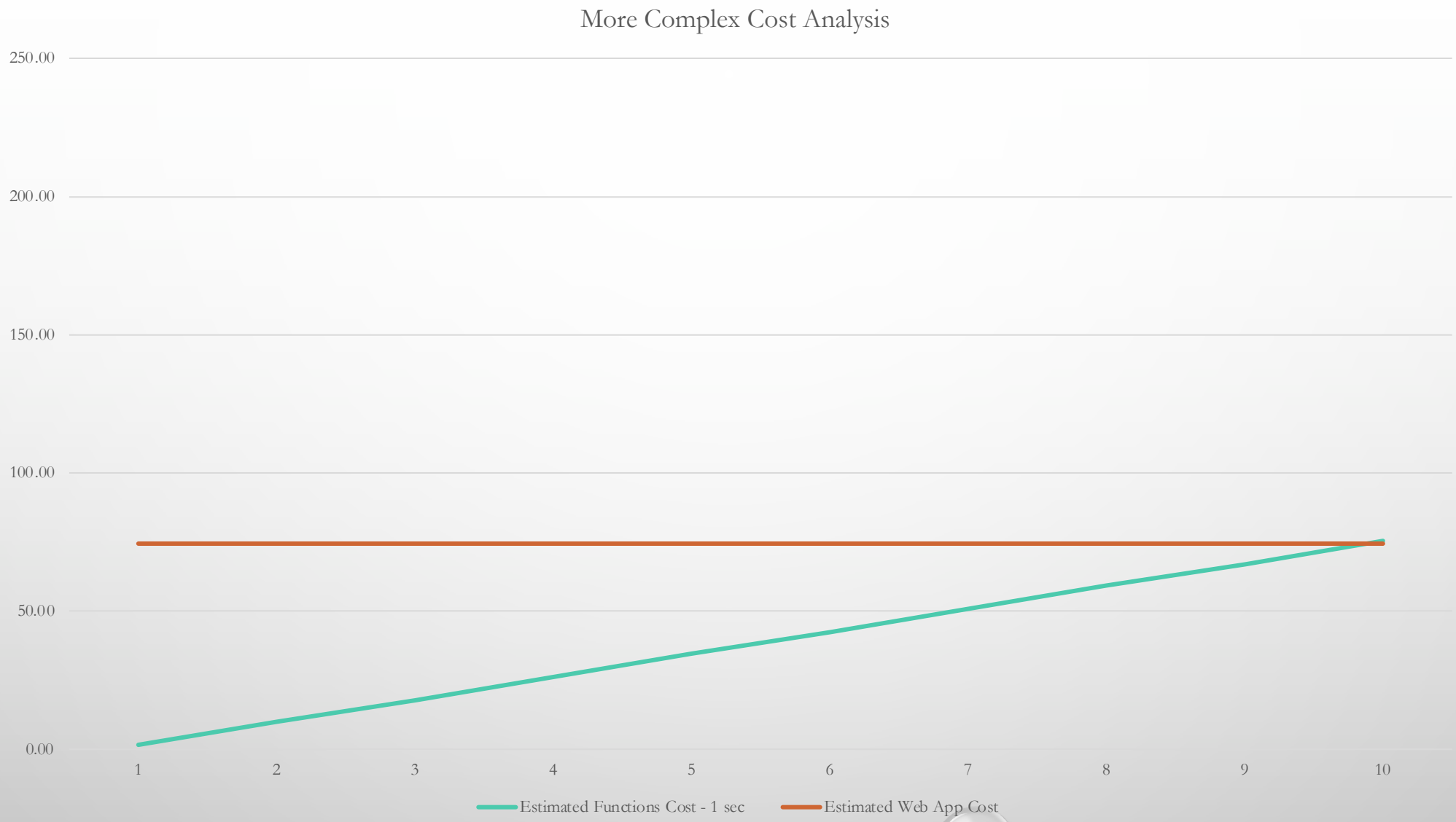


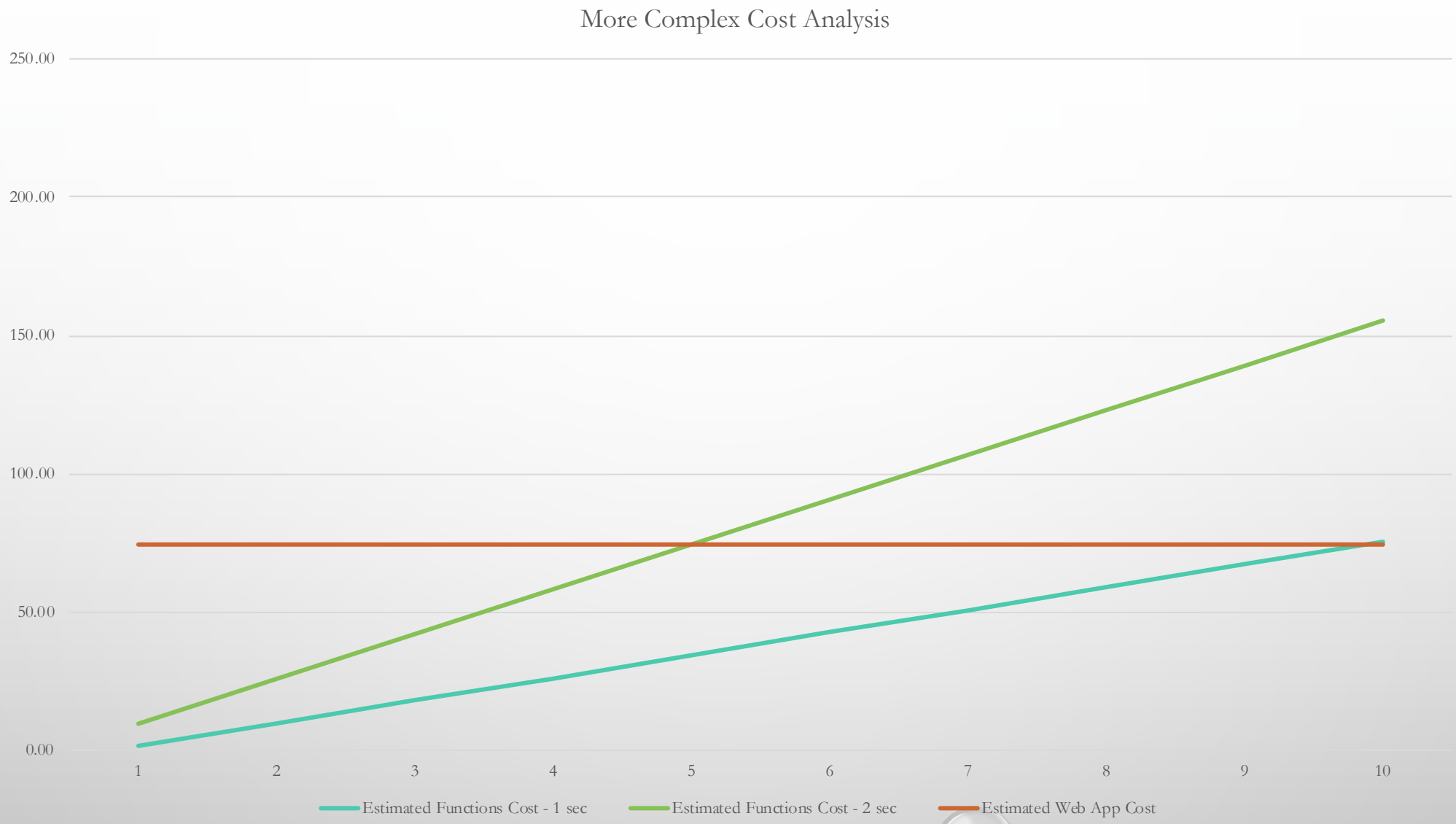
Azure Function pricing

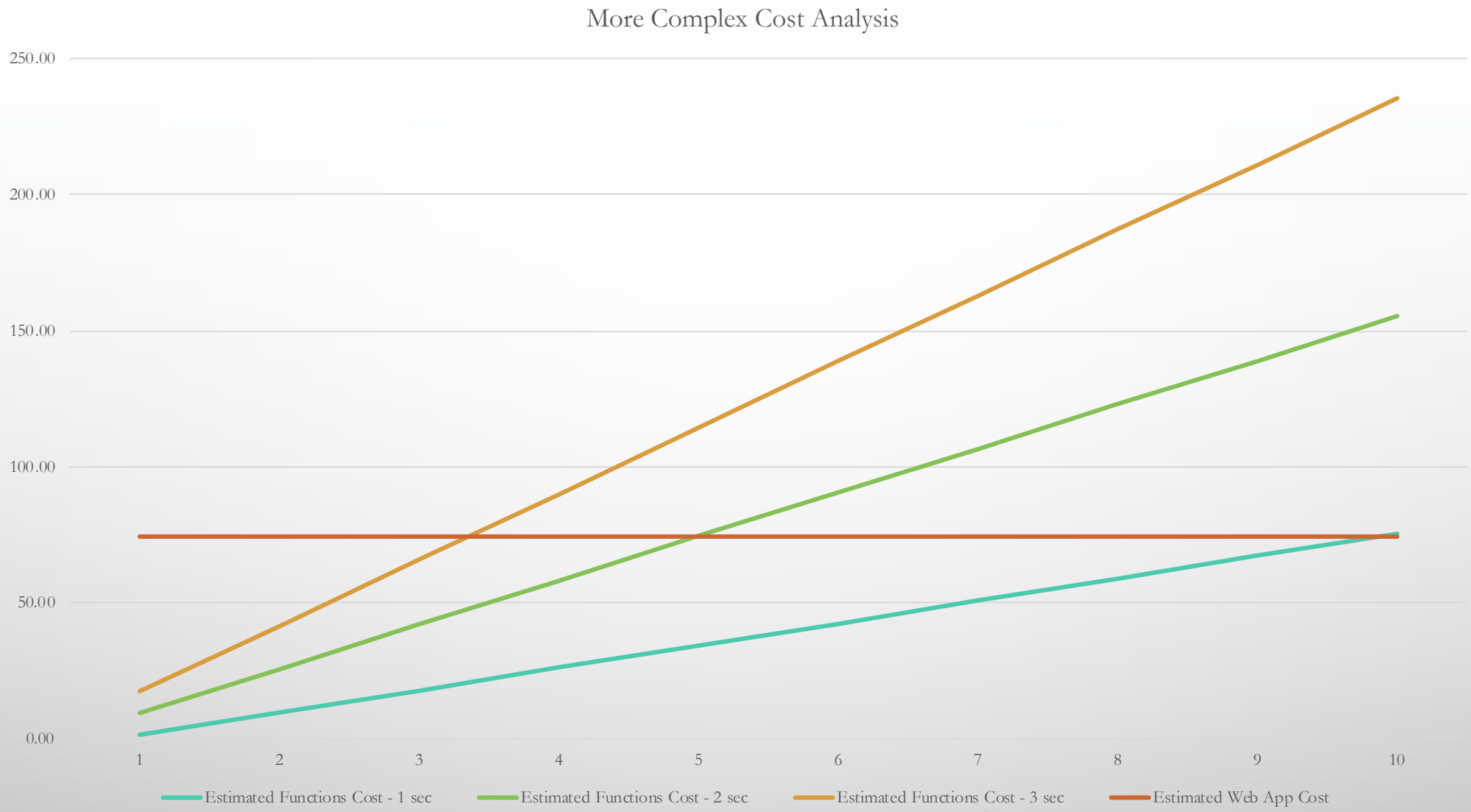
- # Execution * seconds
- Execution memory use in GB/s
- Any storage

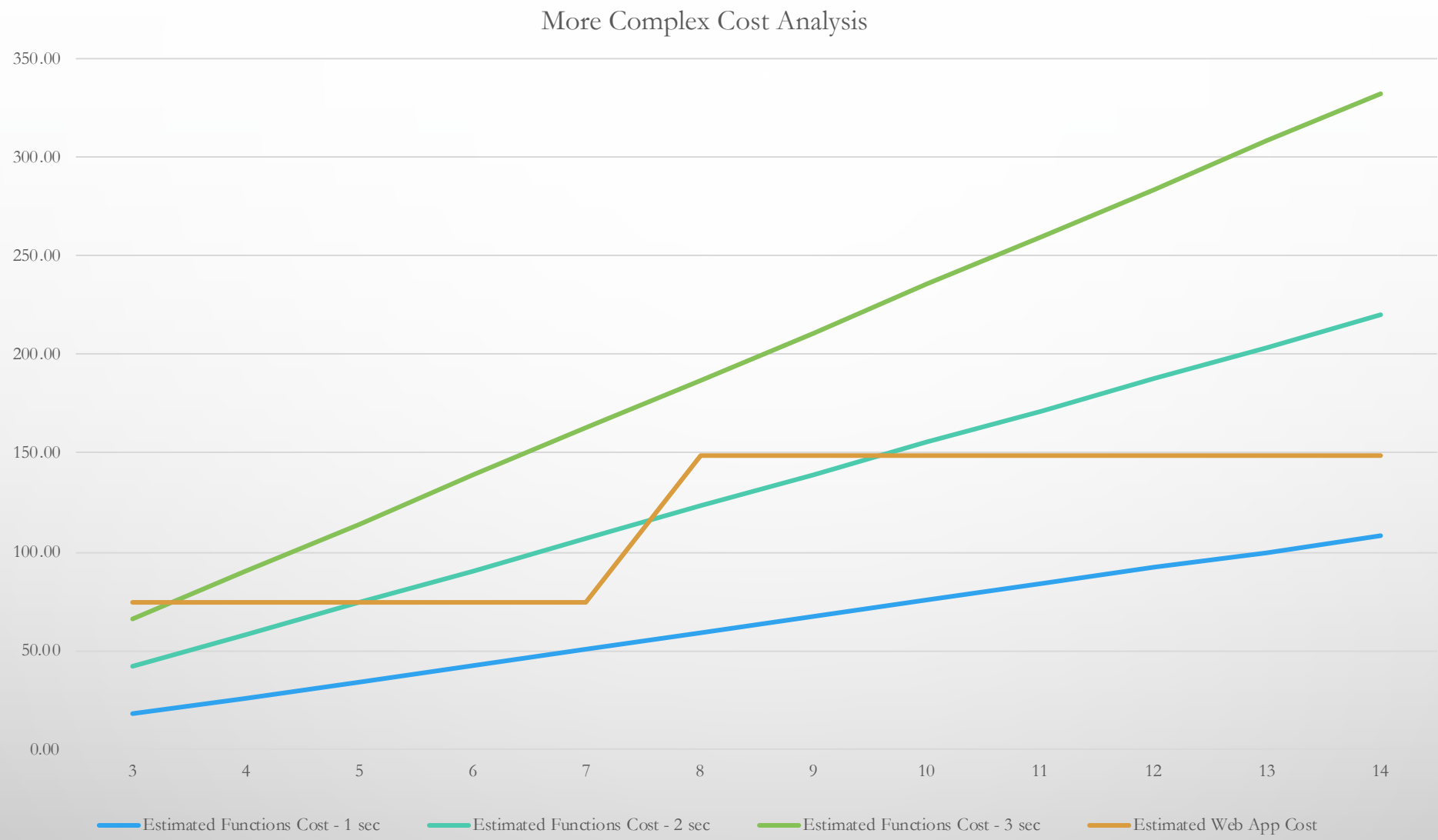
Assumptions:

- Very simple API – 512MB
- Standard Web App
- Function with standard consumption plan


More Complex Cost Analysis
250.00
200.00
150.00
100.00
50.00
0.00
1
2
3
4
5
6
7
8
9
10
Estimated Functions Cost - 1 sec
Estimated Web App Cost


More Complex Cost Analysis
250.00
200.00
150.00
100.00
50.00
0.00
1
2
3
4
5
6
7
8
9
10
Estimated Functions Cost - 1 sec
Estimated Functions Cost - 2 sec
Estimated Web App Cost


More Complex Cost Analysis
250.00
200.00
150.00
100.00
50.00
0.00
1
2
3
4
5
6
7
8
9
10
Estimated Functions Cost - 1 sec
Estimated Functions Cost - 2 sec
Estimated Functions Cost - 3 sec
Estimated Web App Cost


More Complex Cost Analysis
350.00
300.00
250.00
200.00
150.00
100.00
50.00
0.00
3
4
5
6
7
8
9
10
11
12
13
14
Estimated Functions Cost - 1 sec
Estimated Functions Cost - 2 sec
Estimated Functions Cost - 3 sec
Estimated Web App Cost

Azure Functions vs Web Apps - pricing

- Pricing is complicated, but in all cases I have seen it really does match what you are using. There are not any gimmicks or short cuts that I have seen – but you must continue to analyze it and make adjustments.

- Pro tip #1 – run a simulation of both of them!
- Pro tip #2 – always abstract your business logic out of the root level of service, so you are not locked into the approach itself.

Azure Functions vs Web Apps – developer experience

- They are both awesome in either Visual Studio, Visual Studio for mac, and Visual Studio Code.
- Two highlights that I will mention
  - You can run an instance of Azure Function locally and test/debug it as it would be running in the cloud.
  - The Javascript / TypeScript experience of writing an azure function in VS Code is pretty sweet.
- Overall – no real difference

Azure Functions vs Web Apps – Architecture Thoughts

- Common themes in the industry when working in larger domains
  - DDD – isolate your system into bounded contexts which isolate areas of the system that might change together
  - Dev Ops – organize components into groups that need to be deployed together
  - Microservices - multiple teams work on independent services, enabling you to deploy more quickly — and pivot more easily when you need to
  - Self Organizing Teams –teams do not wait for management to direct them in order to increase agility for solving problems independently need to own related sections of a platform
- These themes recommend having an architecture that is broken up into smaller deployable units (change together) that can be managed, versioned, released individually in order to increase internally efficiency and quality.
- Azure functions and the whole serverless movement were built with these trends in mind.

Pro tip - Don’t lock in when you don’t have to*

Make a choice quickly and leave your options open to change.

- Do not couple your solution to the exact features of the technology. Isolate the connections to anything that you might need so that there is very little change required in order to move across technologies – unless there is a major benefit.

- In the end it might make sense to try both out; or change in the future when some of your parameters or the providers approach changes.

* - there are times when it is beneficial to lock in, but with services like this that are still changing very frequently – it does not make sense to lock in prematurely

Use Cases

- A business scenario that you might face
  - No brainers
  - For most cases you should
  - It really depends…so do a tradeoff analysis

No brainer scenarios

- You have a medium volume, low to medium complex website that needs a simple back end with a lot of APIs – but the back end relies on another service that might be slow
- You have an existing web site on App Service and have a small function that needs to run on a daily basis to summarize some data and cache it
- You are leveraging a 3rd party API that you need to wrap some simple business logic around per call

Scenarios where most likely is x, but you should review

- You have multiple complex front ends built with a modern js framework, this application requires a low complexity api that should get moderate usage, managed by a single team that would like to release everything together

- You have a monolith application and are carving off the first service which is small and fast into a micro/macro service

- You have an issue in your system and you need to have something that runs every day to fix it until you get the real fix in (which could take a few days or weeks)

- You have a data integration that is customer specific that you would like to execute and bill the client explicitly

It depends - do a tradeoff analysis

- You are building an API which will vary in volume, maximum performance is key but cost is a major consideration point
- You have a monolith application and are carving off the first microservice and it is a highly complex and time sensitive api
- Posting a file that will kick off a data integration or internal process

Any time where good enough is not good enough OR you have competing requirements

My Opinionated Guiding Principles for technical decisions

- Don't make one size fits all decisions in your organization –slow down just a little and make an informed decision.
- Defer decisions as long as you can and leave flexibility to change
- If you don't know – try them both out – if designed correctly, you can easily switch between these features and do a comparison on performance / cost / etc. Limit getting locked in, unless there is a key business advantage.
- Only use a weighted scale when you need to – you and your teams should be able to logically discuss what is best for the business
- Do not force unilateral decisions across teams – you will lead to broken systems and to teams that don't understand the options and the reasons for decisions.

The beauty of deferring decisions

- Wrote an application (THE BUTTON) that migrated data from a VB6 application to a modern web front end / api system
- There was a very complex data mapping process that would occur between
- We did not know how much this would be used and how the users would react to the time it took.
- We wanted to get it out to the market as quickly as possible to learn.
- #1 we built it using 2 azure functions.
  - 1 as a http where the VB 6 program could upload a request and put it in a queue and storage
  - 2 as a timer function that would be able to process all waiting requests.
- If volume was large this would allow us to use a queue trigger and handle these concurrently
- If the processing time was large we would be able to manage the amount of functions running at the same time
- If we had a lot of technical issues with the dependent systems we could add in logic to retry and reprocess items

<u>In the end, we did not need any of these because it was fast enough and the volume was spread out well – but we had options!</u>

Thank you

"If anyone can refute me—show me I'm making a mistake or looking at things from the wrong perspective—I'll gladly change. It's the truth I'm after, and the truth never harmed anyone." – *Marcus Aurelius*

If you have any feedback – good or bad, I am all ears as I would like to continue to make this presentation better.