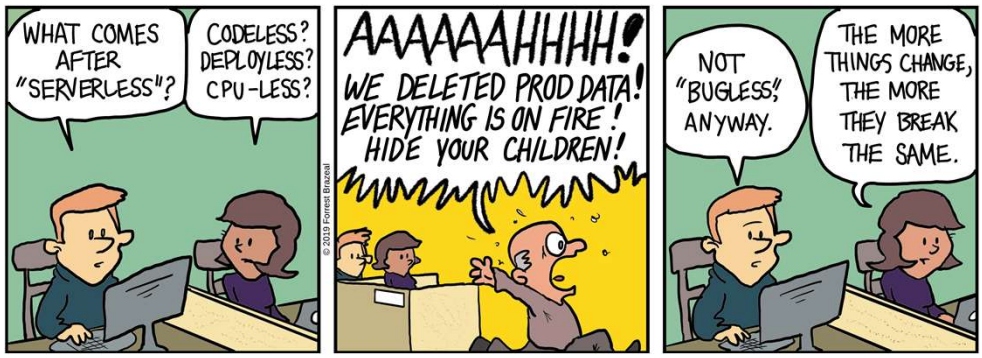
Good afternoon! I'm Lucas Gallagher, Security Architect with Foot Locker,
and today I'll be talking about **securing serverless technologies**.

As the technology that we are highlighting here, let's talk in brief about **[X]**

SERVERLESS?



Forrest Brazeal - https://faasandfurious.com/108

'Serverless' -**serverless compute** in a nutshell is a platform that allows code or operations to be performed **without knowledge of underlying infrastructure** w/o having to worry about or maintain **hardware**, **operating system**, **dependencies**, or even **software** in many cases.

Popularized by **AWS's Lambda**, allows code->upload->Lambda func->run w/defined **trigger**. 'Functions as a Service' or **FaaS, and all the clouds are doing it these days**. Google Cloud - Cloud Functions, Azure - Azure Functions, etc. Extends into many **other domains** as

well: DBaaS, low/no code platforms (Azure's Logic Apps), and others

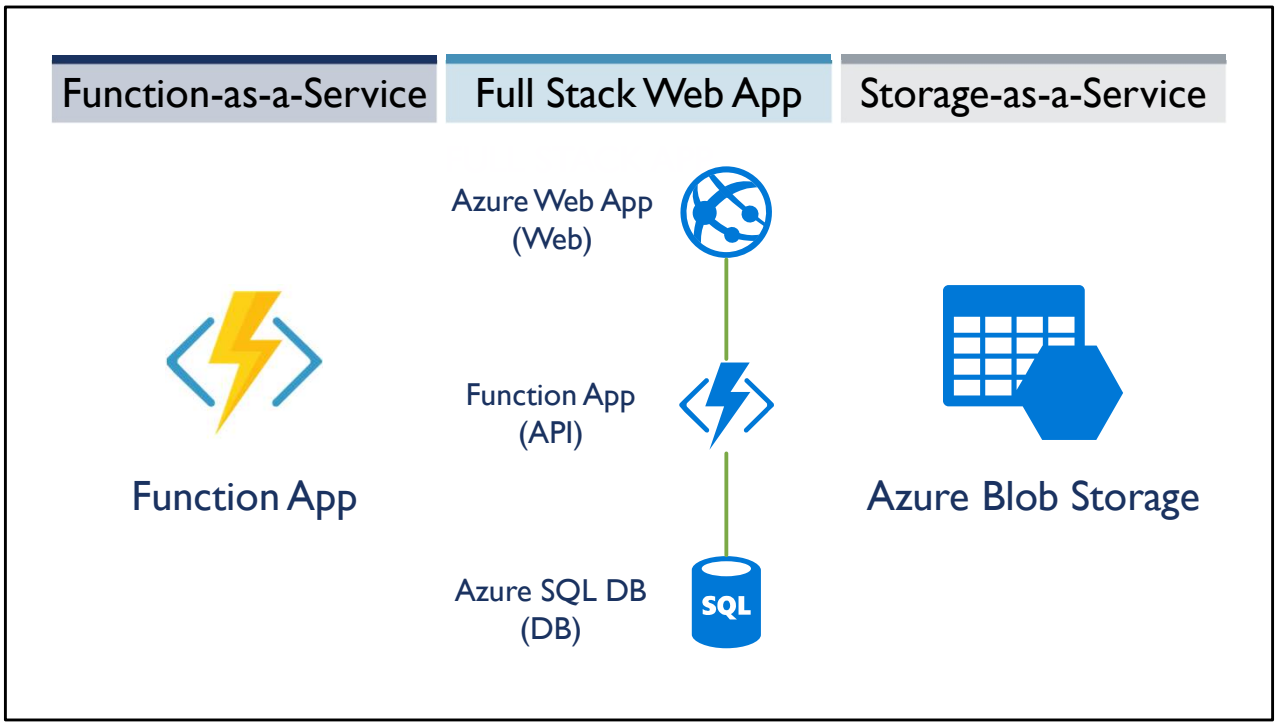
Powerful stuff! In some ways this is **fulfilling the promise of cloud computing**--not just 'my VM in someone else's datacenter', but 'my application runs on the cloud'.

The comic pokes fun at it, (issues), but with serverless technologies it becomes much easier to create

- **automation**,
- spin up **database**
- a full stack application

with **shorter time to market** that has greater **capability** and **integration**.

[X] Here are specific examples in Azure.



[X] Azure Functions is a serverless compute platform. It can run a single script on demand, or a fully defined API for an application.

[X] Building on that, the second example is a simple **full stack application** deployed in an Azure subscription. The web frontend? Serverless. The API? Serverless. The Database? Serverless Database as a service.

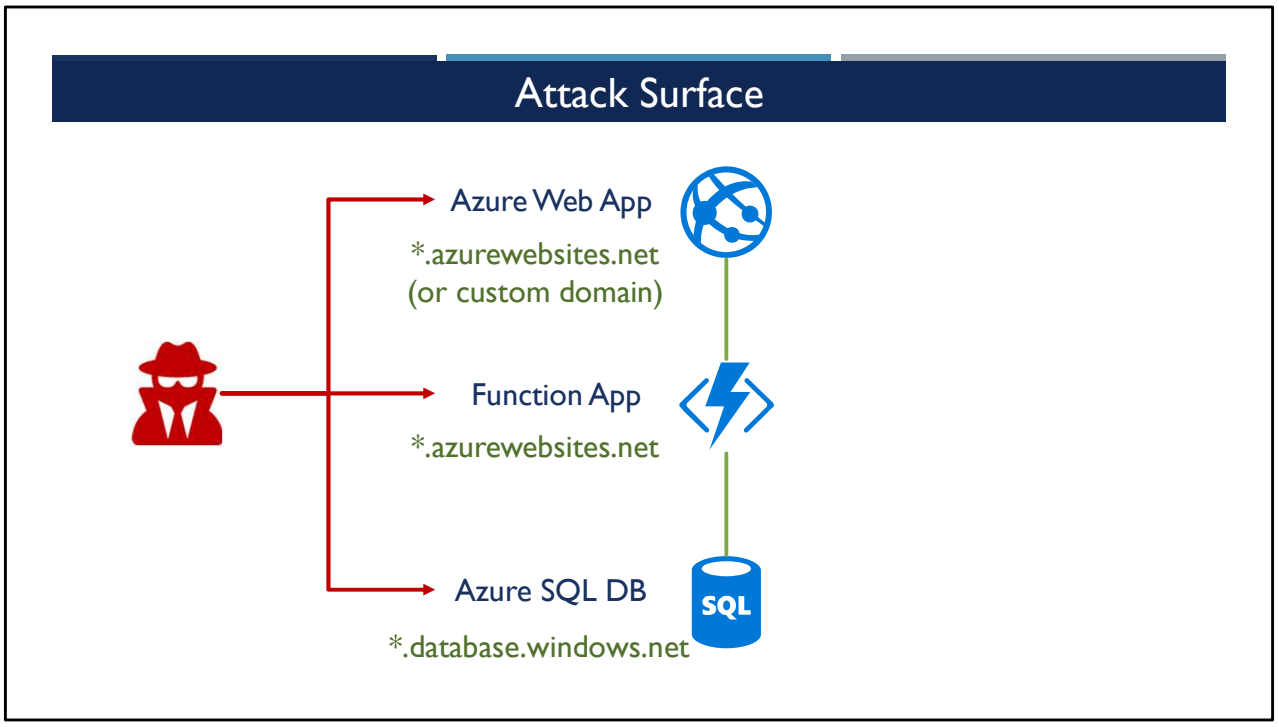
No element of this requires managing a server or a host. Easy versus traditional And it is relatively easy to assemble and deploy--especially when compared with the learning curve of deploying a full stack app with other methods.

[X] Lastly we have **Azure Blob Storage**--think Amazon S3 buckets. Could drag and drop files using a desktop app, or it could be the storage backend of a complex system.

All of these can be interacted with in various ways: **web ui**, desktop client or IDE extension, via API, or directly via a URL.

As you can tell, I'm a fan. Alright, done geeking out about serverless. Hopefully I was

able to convey the **benefits and possibilities**, and why these technologies are in many cases **transforming business applications and processes.**



Sticking with web app -> attack surface. **Attack surface is your exposure to threats.** Can involve unauthorized access, denial of service—all the things that can compromise the CIA of your app, data or environment.

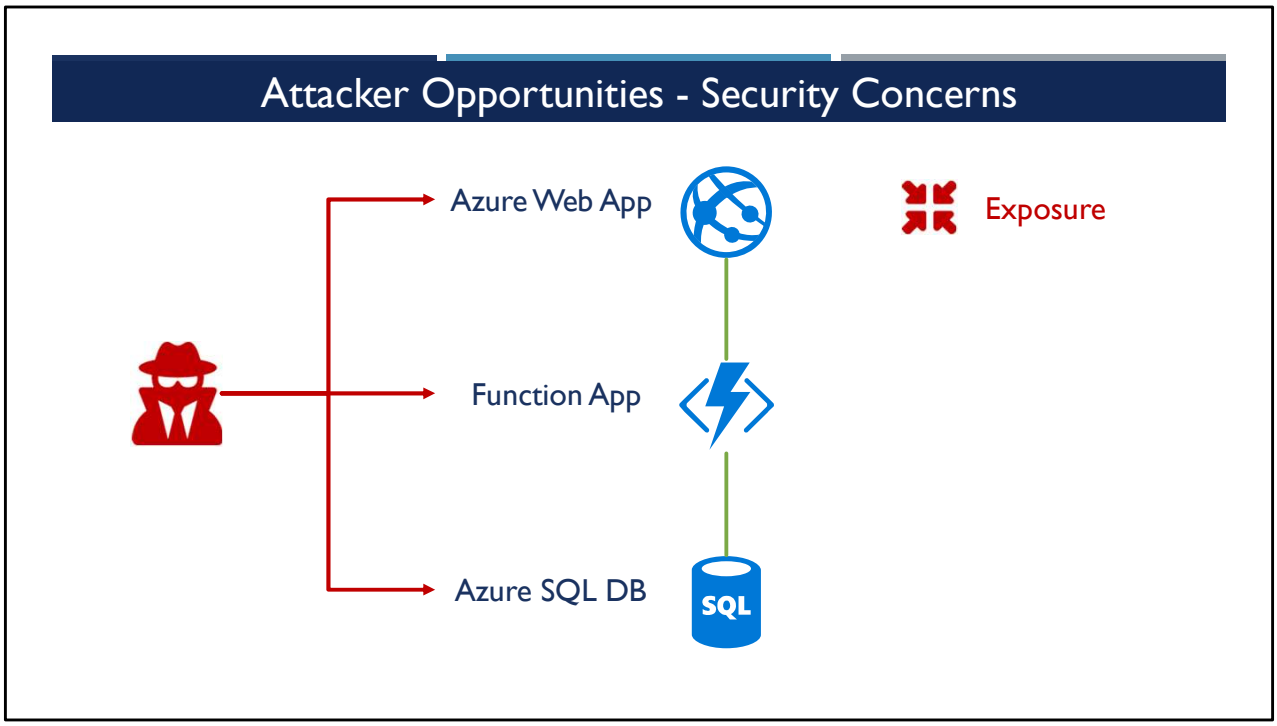
Physical locations -> armful of donuts corp site. Virtual locations -> server with a port accessible from internet

couple ways Serverless ^ atk srfc over traditional—including one way that differentiates vs server compute resources in a cloud such as Azure.

Unlike traditional applications, where just the **webhost frontend** is internet-accessible, a deployment of most serverless entities are all internet-accessible by default--
[X] and it's **not a bug, it's a feature!**

One of the great benefits of Serverless offerings is how flexible and interoperable they are. A function can operate standalone, or as part of a larger full stack application. But that same flexibility means that it is really easy to accidentally--or in ignorance--deploy it in an insecure manner—
[X] and with a large attack surface.

Let’s dig a bit deeper into the opportunities the attacker has—and conversely the security concerns around serverless



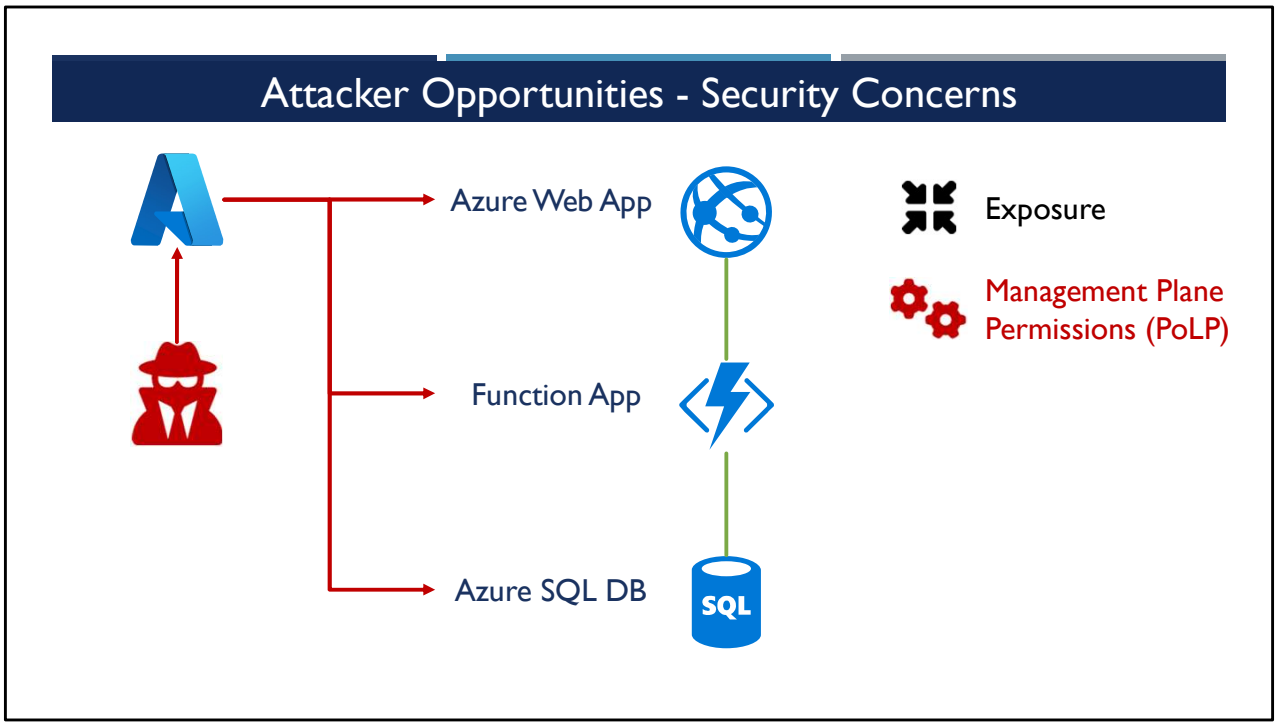
Exposure

Can anyone or anything **outside the company** access or trigger operations on these components (and is that access appropriate to the function (pun intended) they are fulfilling)?

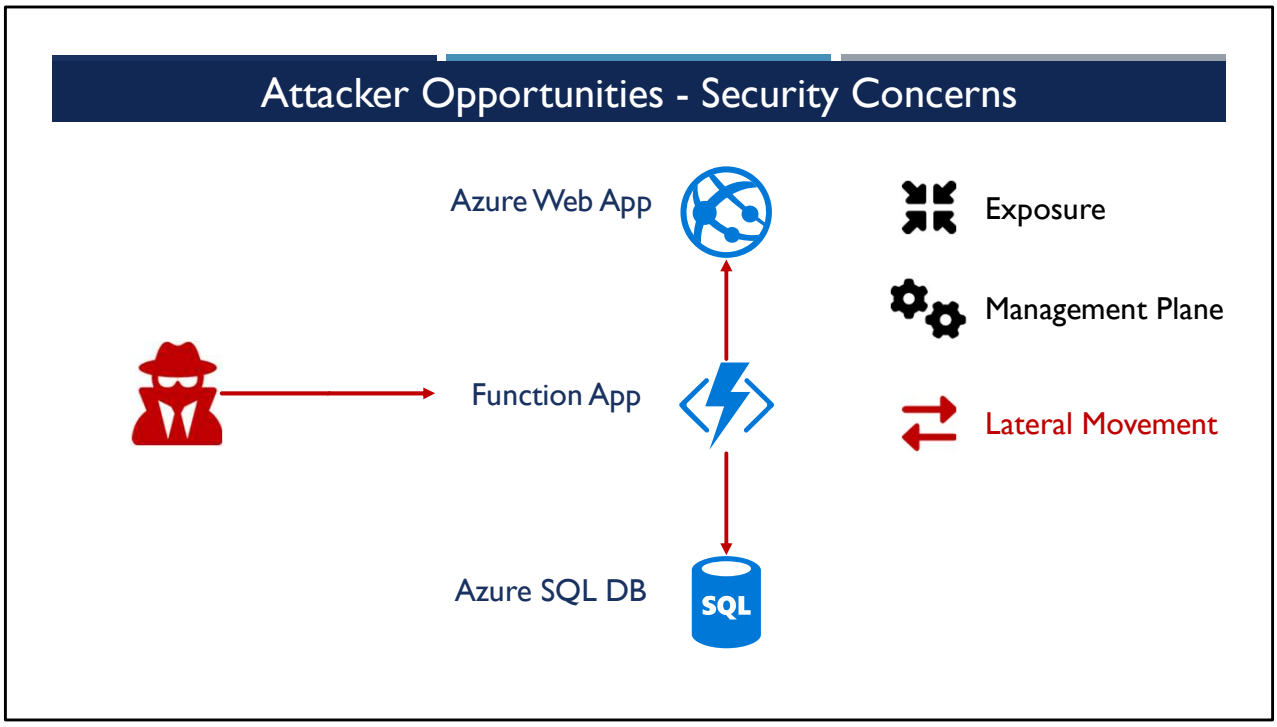
- Can anyone or anything **inside the company** access and trigger operations on them (and is that appropriate)?
  - If a systems engineer unrelated to the application can access the contents of that database
- **Where appropriately exposed**, have it been validated validated whether there are vulnerabilities in the logic of a given component?

- Does that function app accept input with SQL Injection included?

[X] Other attacker opportunities/security concerns

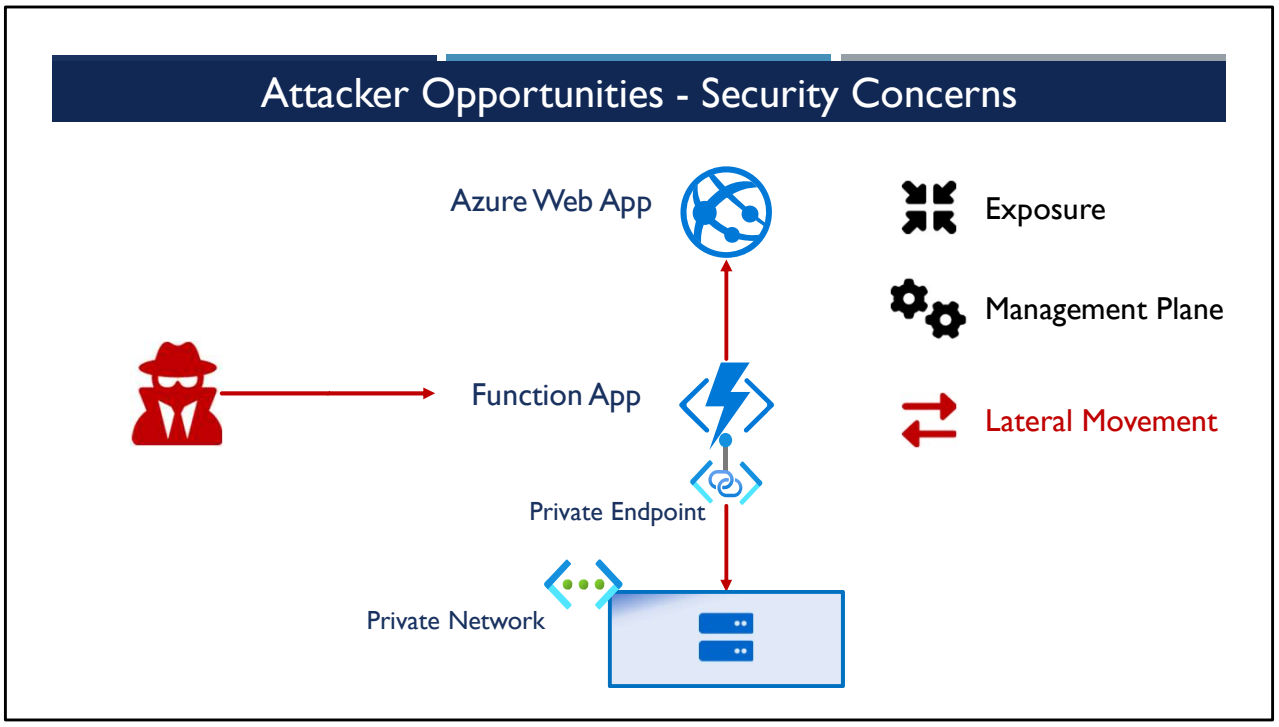


Management plane—as with all resources living in a cloud provider, if your management plane is compromised, or access is achieved with an account with the right (or wrong permissions), everything else will follow. The Principle of Least Privileges applies here: access should only be granted to those who need to have it.



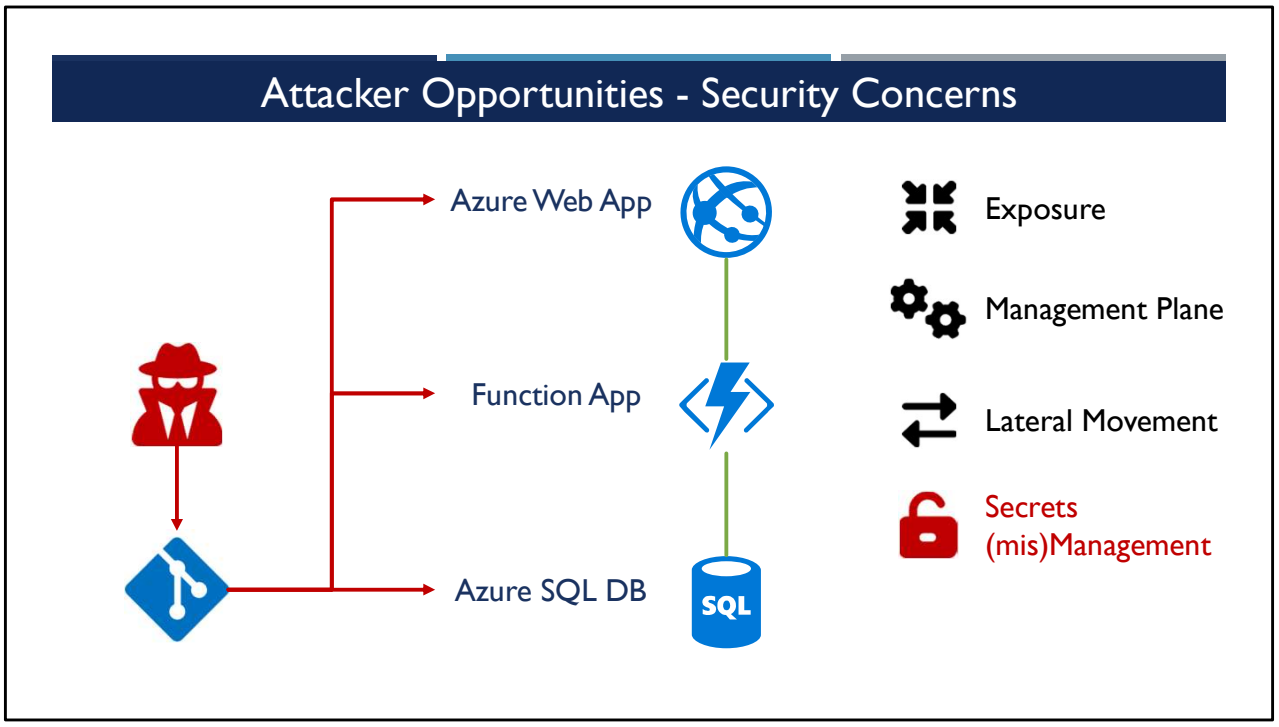
What can the app itself access? If someone was able to get the app to execute their code, what could it reach in your exposed services with inherited permissions (like in that DB-as-a-Service)?

How much scarier does this same scenario become when this application architecture looks more like **[X]** this?



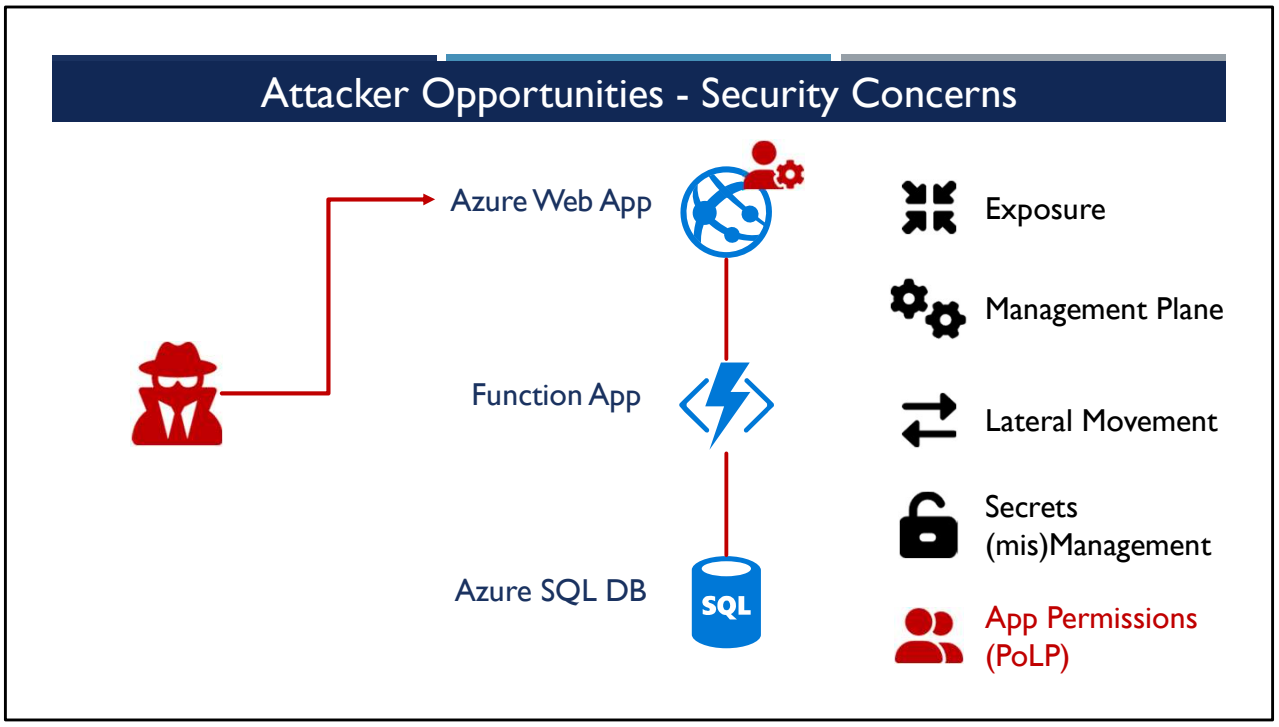
[X] this? Here we have serverless components connected to your private network.

If this web app was stood up, either separate from or joined to the rest of your envinronment, would you detect it? Be able to prevent it? Validate the that it was properly secured? Detect when malicious traffic was moving internally?



Are secrets kept in code?

I love serverless, so I talk about the benefits a lot. One of those benefits is how simple it is to configure and deploy using code. In the case of an Azure Web App you can literally, from the portal, point to a GitHub repo and it will provision the build process to pull the code and populate the web app. What if that is a public GitHub repo? And connection strings for the DB are in the code, or secrets for the API?

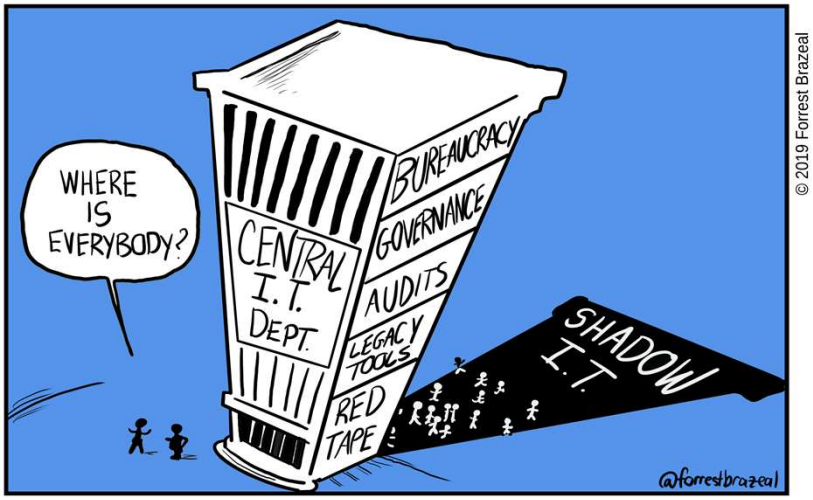


A web app or function may **appropriately** be capable of doing a lot--but should **everyone** who can operate that web app be able to do everything? Does your web app have roles included?
Back to **principle of least privilege**

You may be thinking 'wait a minute, those are all issues for normal, server-based web apps. This is all **secure application development 101**'--and you are right– (though again serverless component in isolation = more attack surfaces.

What sets serverless apart isn't the types of vulnerabilities--it is how easily serverless can be deployed w/o governance. As a service. And as with anything lacking governance, we are now talking about **[X]** shadow IT.

Shadow IT



Forrest Brazeal - https://faasandfurious.com/99

We've all heard the term 'Shadow IT'. It's **unmanaged IT**--unmanaged technology. And Shadow IT represents the worst kind of attack surface—the kind you don't know about.

used to be **Desktop** an employee brought in to play music for the office. Plug it in - **an ethernet wall jack:** they get music, you get an unmanaged device on the network.

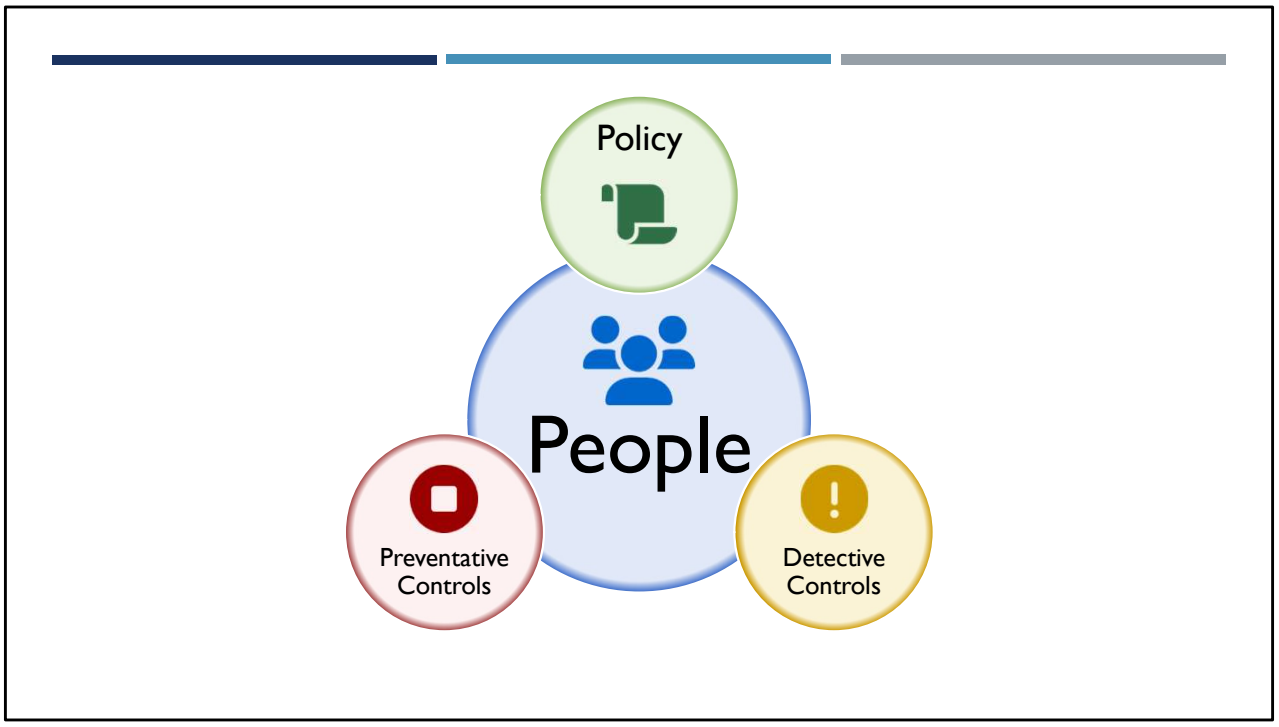
now can be a **dept w/credit card** starting an Azure sub, putting company data in a SQL DB. Or an existing Azu

sub connected to your on prem, with an engineer who creates and hands off a function app for a dev to play with this cool new tech.

New technology brings new security concerns—new attack surface. Even if it is just in the form of new avenues for the same old techniques. So for a biz/non-profit **when New Tech, what are the options?**

Prevent (new, not yet used, any use = shadow IT so prohibit)
Implement now, figure it as you go, pay the piper later. Security comes after. After five departments deployed serverless in 3 different cloud providers, someone (hopefully someone internal to the org and not Brian Krebs). Then there is the scramble to develop a standard and everyone undergoes painful process of aligning to the standard along with all the rework that entails (and that is frankly a best case scenario, because the alternative is the risk of missing something that could lead to a breach).

Secure by Design - security (whether dedicated **personnel** or simply **considerations**) is part of the adoption of a technology, and secure standards, design patterns, and templates are created
[]

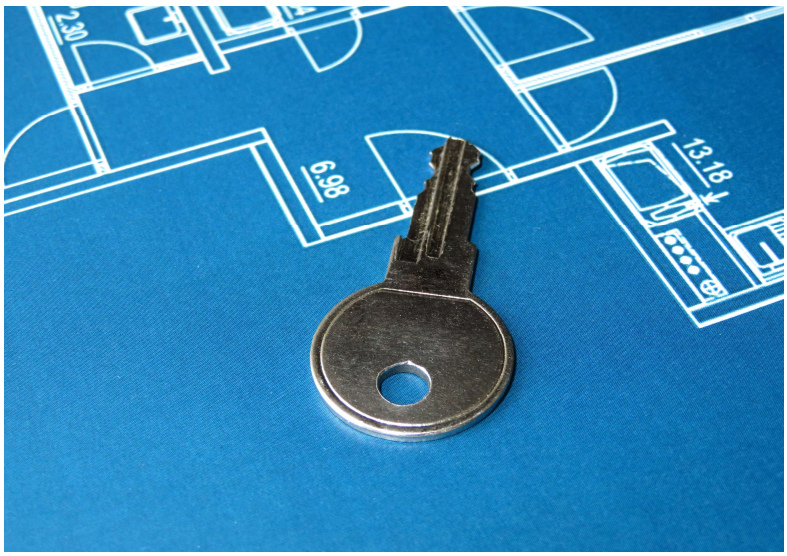


Hear me on this: as a business or nonprofit, you need to get ahead of new technologies to reap the benefits and avoid the risks—and people are the key. you need either **technology-minded security** folks, or **security-minded technology** folks. A group who evaluate the new tech for your company, with the passion, ability, and time to think ahead. This could be a practice area, center of excellence, guild, tiger team—whatever the term you want to wrap around it.

They need to create governance **[X]** around these technologies: what is appropriate use, what is misuse,

define what appropriate use looks like

and how inappropriate use will be **[X]** prevented or **[X]** detected. Policy and Controls to address each of those attacker opportunities, and reduce or mitigate your attack surface. Not blocker, AND to help with adoption! To provide guidance, templates, example deployments to let serverless have the greatest impact and benefit possible. **Serverless is awesome. Make sure it serves you, not your adversaries.**



https://github.com/epopisces/talk_attack-surface-as-a-service

THANK YOU