


# Are we getting infrastructure all wrong in the Serverless era?

 Yan Cui · Follow  
Published in theburningmonk.com · 4 min read · Jun 30, 2023

👁 72    💬 6    📌    ⌚    📄    ⋮



During my last conversation with Ben Kehoe on the [real-world serverless podcast](#), he said something profound and it's been bugging me ever since. And now I'm ready to infect you with this brain bug! You're welcome ;-)

On the infrastructure-from-code trend, Ben said:

"I think infrastructure-from-code is wrong on all counts. I don't think we should call it infrastructure... (skipped)... When we talk about business logic, **we missed that business logic exists in things that are not general-purpose programming languages**. The choice to have an SQS queue in your architecture is a business logic decision. It's saying that I need persistence for this message that is passing through... (skipped)... Similarly, I'd like to say that least privileged policies is business logic because the choice of what matters and doesn't matter in an authorization decision, is specific to the business case you're doing... which is the definition of business logic — the irreducible complexities that are specific to the problems that you're solving."

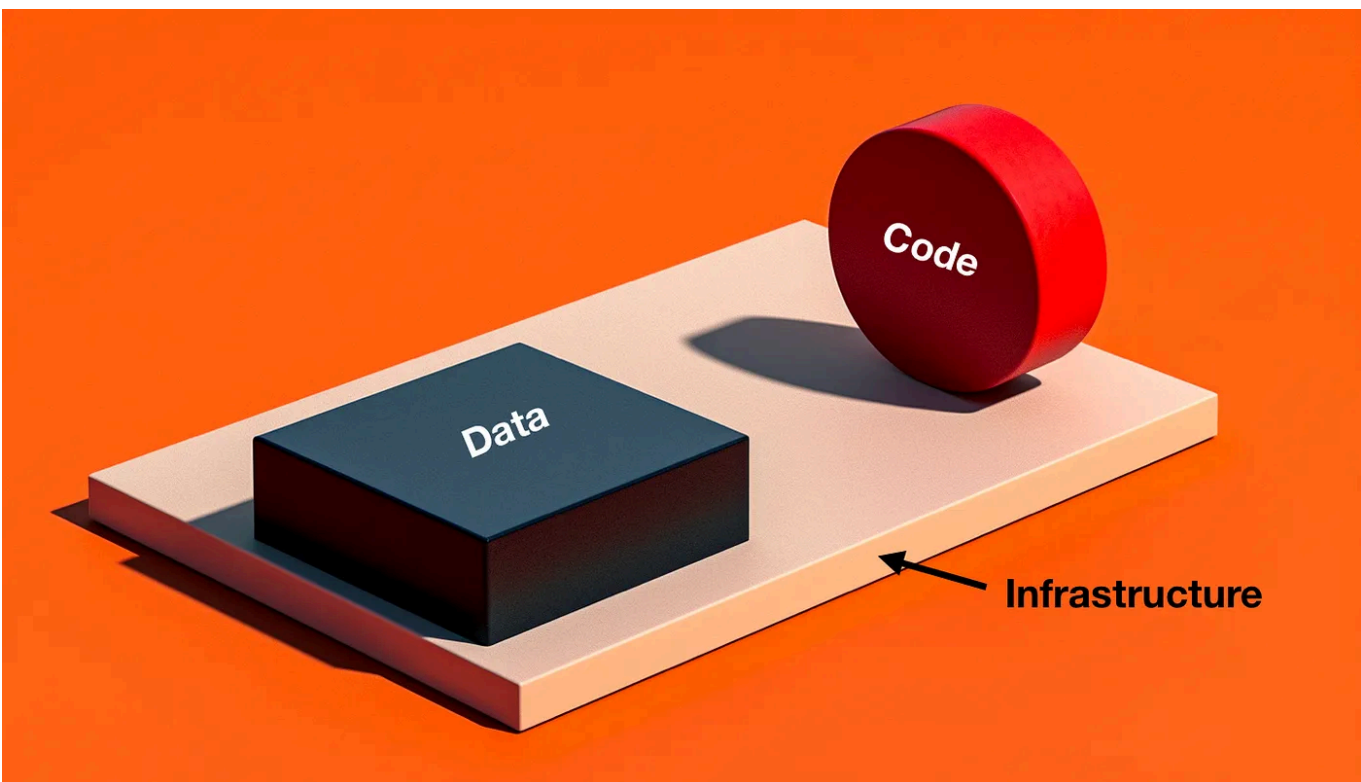
(you can listen to the whole conversation [here](#))

As Ben points out, choosing one service (SQS) over another (SNS) is down to business requirements, as are the authorization decisions in your application. This angle of looking at serverless architectures really blurs the line between infrastructure and business logic.

🔍 Search

✍ Write    🔔    

In a serverful application, you deploy your code onto some kind of a server (be it virtual machines or containers). And often we also deploy other software, such as databases, onto these servers. In cases like this, it's easy to tell the difference between infrastructure and business logic.



In a world where your primary interaction with AWS is to “deploy software onto servers”, it's understandable that AWS resources are synonymous with “infrastructure”. After all, you are not using the native services that AWS provides and are only using the infrastructure-as-a-service parts of AWS.

This way of looking at AWS has carried over to the serverless world. And the fact that the tools we use to develop our applications are called “infrastructure-as-code” (IaC) further enforces this view. That any AWS resources that we create with these IaC tools are “infrastructure”.

But as Ben points out, many of the architectural decisions we take are specific to our business and determined by our business requirements. Which messaging service do we use, do we go with EventBridge or SNS or Kinesis? How do we integrate them together, do we write Lambda functions or use direct service integrations? Who is allowed to access the system and what actions can they perform?

You can implement a CRUD endpoint by writing a custom Lambda function or have API Gateway access DynamoDB directly. One involves custom code and the other just a few lines of configuration in your preferred IaC tool. So does one count as business logic and the other as infrastructure? Even though both amount to the same result?

And what if you use IaC tools that use general-purpose languages instead of YAML? Is “declared in YAML” the definition of “infrastructure”? Does something count as “business logic” only if it contains custom code instead of “configuration”?

For a long time, I've felt this awkwardness when talking about infrastructure in the context of serverless. Because it's not always clear to me where infrastructure stops and business logic starts. After talking to Ben, I still don't know for sure! But I'm sure the distinction is not as clear-cut as we thought and we shouldn't look at AWS resources through an infrastructure-tinted lens.

In the serverless world, we want to leverage the cloud to handle as much of the undifferentiated heavy lifting as we can. And that often translates to using a service as opposed to writing custom code. How we configure those resources and what we ask them to do is specific to the business problems we want to solve. And so, they are as much a part of our business logic as any custom code we write.

Top highlight

So Ben has convinced me, but what do you think? Do you agree, or disagree, or maybe you think we should look at serverless in a whole different light?

Let me know, I'm curious.

AWS    Serverless    Software Engineering    Cloud



Written by Yan Cui

5.8K Followers · Editor for theburningmonk.com

AWS Serverless Hero. Follow me to learn practical tips and best practices for AWS and Serverless.

Follow

