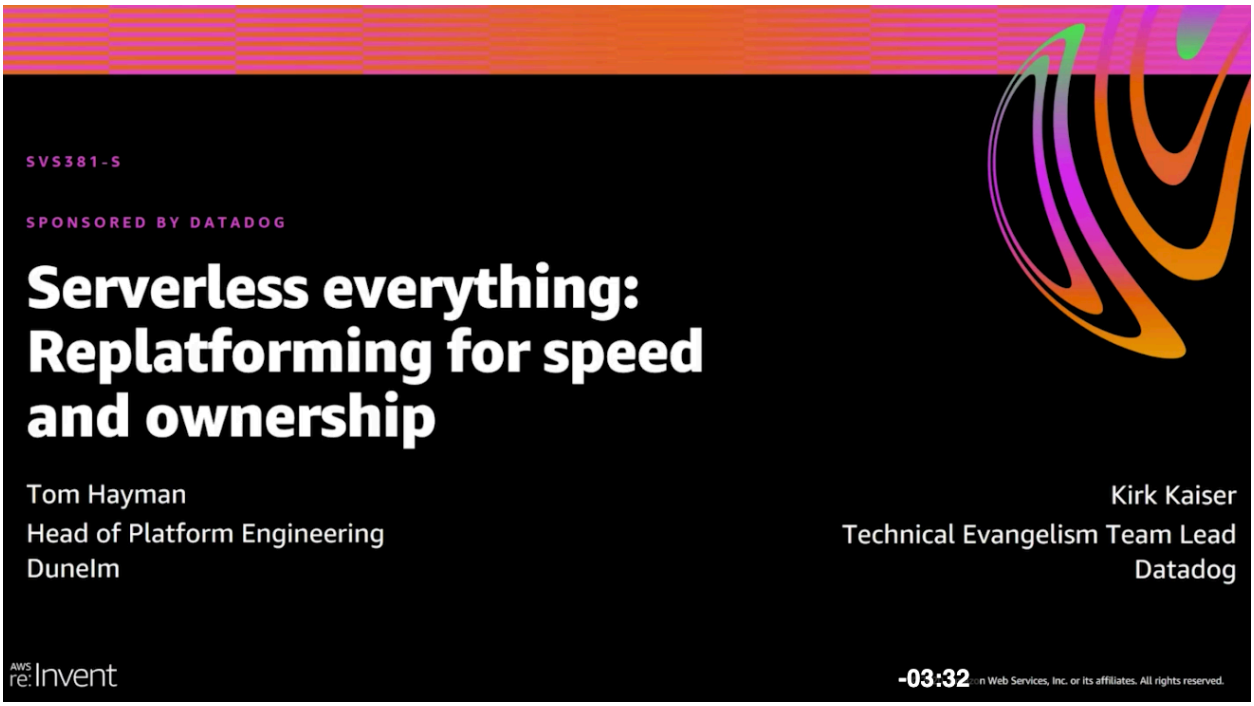
AWS ReInvent

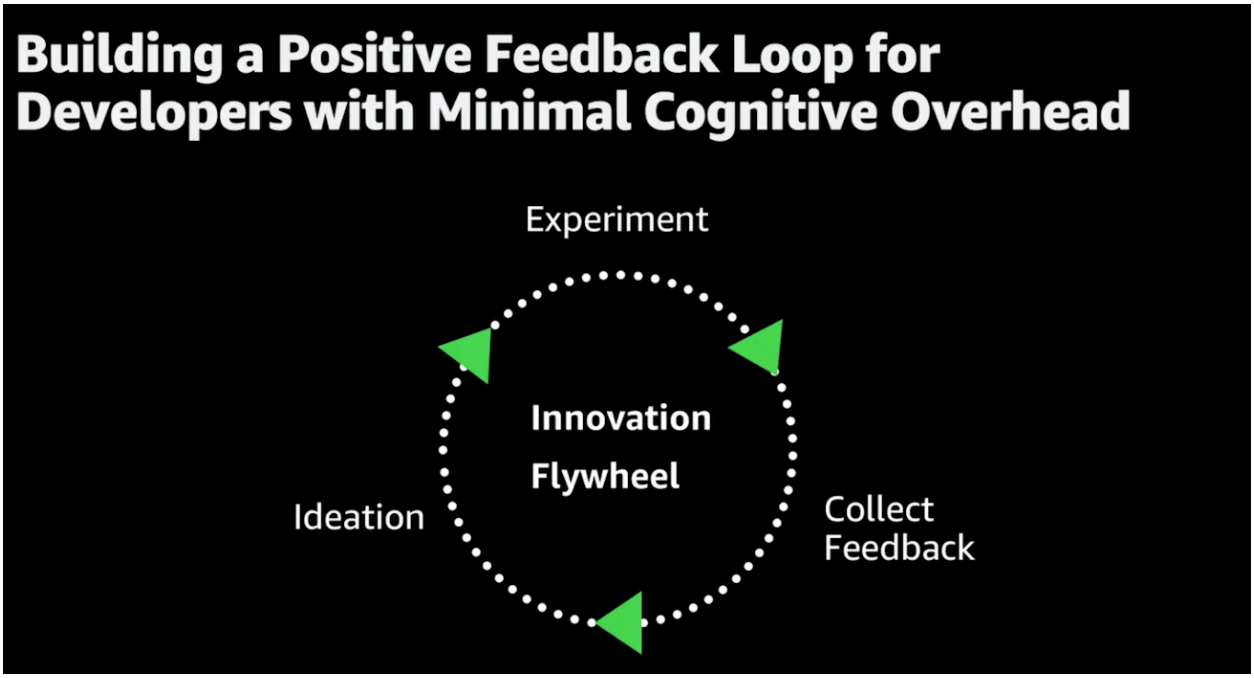
Serverless everything: Replatforming for speed and ownership

(sponsored by Datadog)

Companies all over the world are reaping the benefits of serverless and pay-for-what-you-use compute. Few, however, have adopted serverless to the extent DuneIm has. During a replatform of its online sales platform, DuneIm chose to build a completely serverless microservices application. Hear the story of how DuneIm moved to deploying changes over 200 times a month and saw an average speed improvement of 472 percent across its entire platform. In addition, hear how it has decided to organize teams around core business domains, and see how observability is a core component of moving fast with serverless. This presentation is brought to you by Datadog, an AWS Partner.



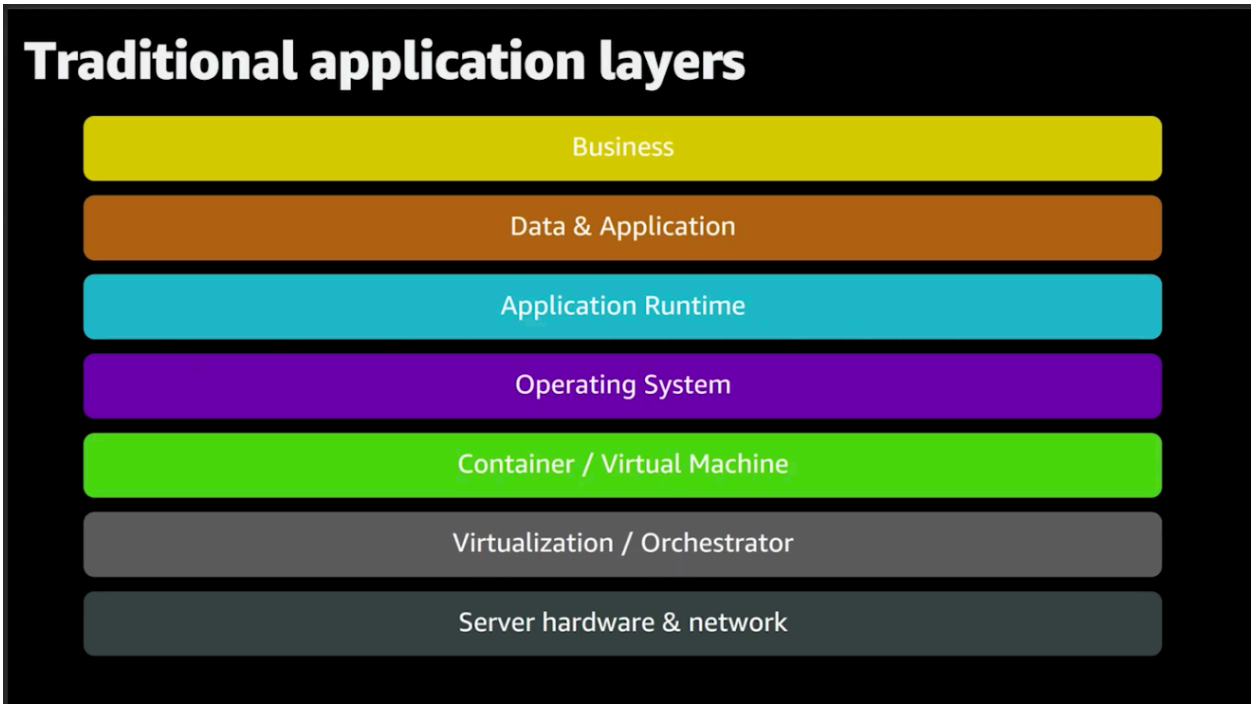
Companies all over the world are reaping the benefits of serverless and pay-for-what-you-use compute. Few, however, have adopted serverless to the extent DuneIm has. During a replatform of its online sales platform, DuneIm chose to build a completely serverless microservices application. Hear the story of how DuneIm moved to deploying changes over 200 times a month and saw an average speed improvement of 472 percent across its entire platform. In addition, hear how it has decided to organize teams around core business domains, and see how observability is a core component of moving fast with serverless. This presentation is brought to you by Datadog, an AWS Partner.



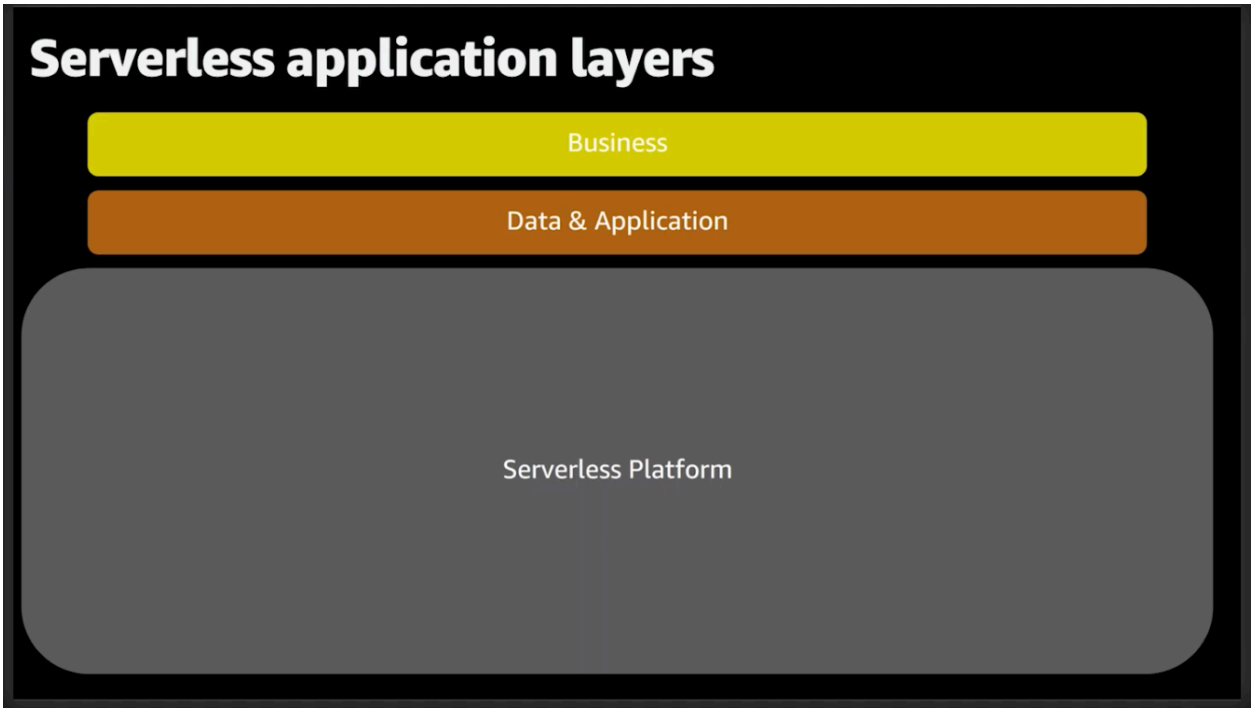
Smaller companies are adopting serverless more than larger companies because they are more nimble.

With Serverless, forced to reason about code in smaller pieces, think less about infrastructure underneath it and more about the direct value of the code you are writing.

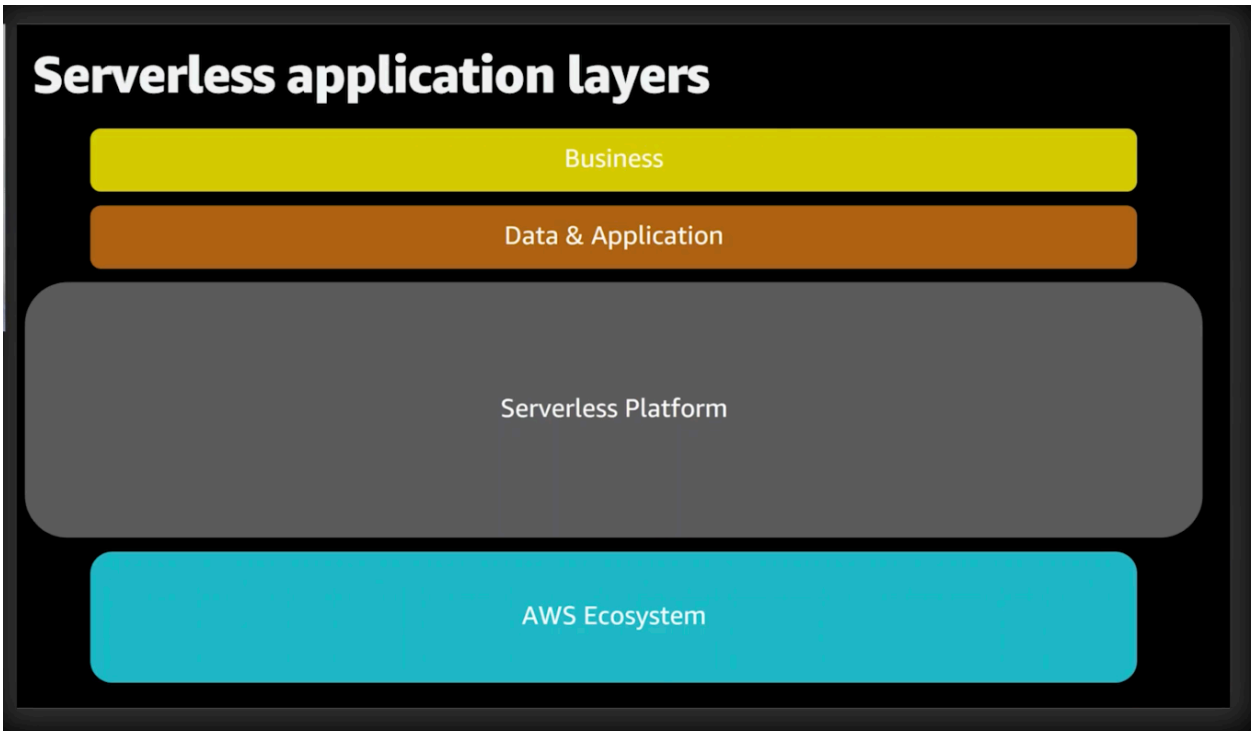
Getting out of loop of existing instructure and how we deploy code.



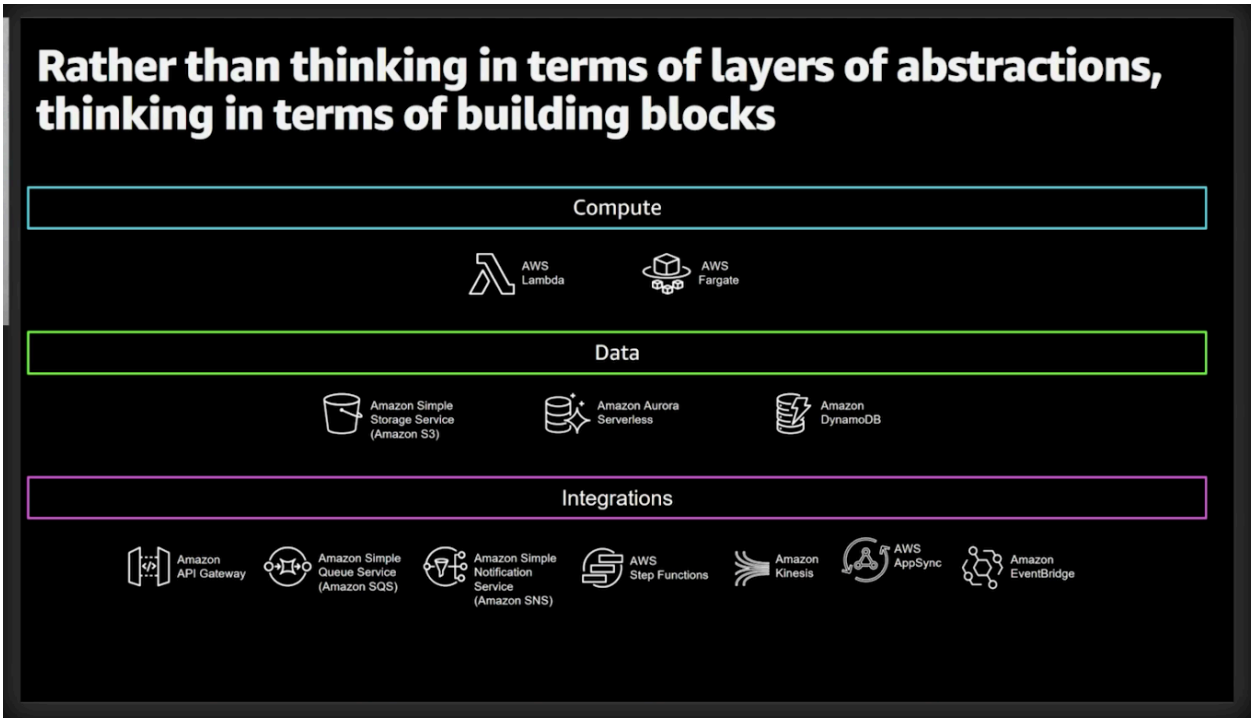
Traditional web app: Have to make decisions about all of these layers. Instead of thinking about what we should be building, we think about each of these layers.



Write smaller units of code that are written independently and deployed on their own. Individual empowerment: don't have to coordinate releases as much, more autonomy.



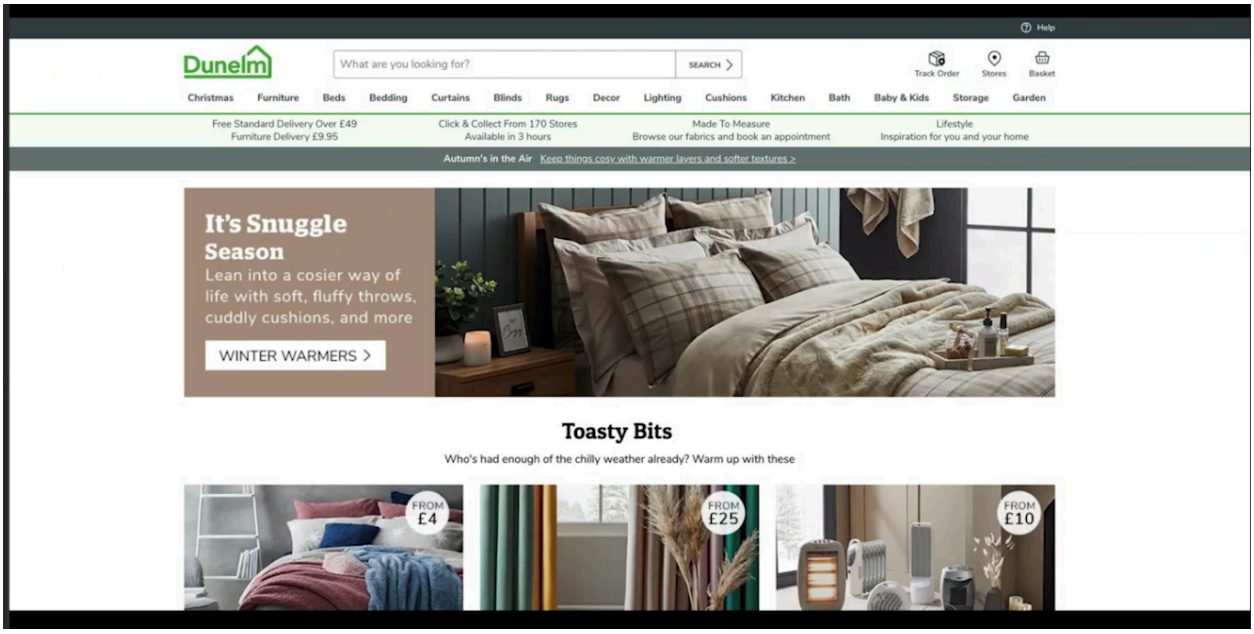
Your code never exists in isolation. Serverless fits into the entire AWS ecosystem. Don't have to think about underlying way they run and are built.

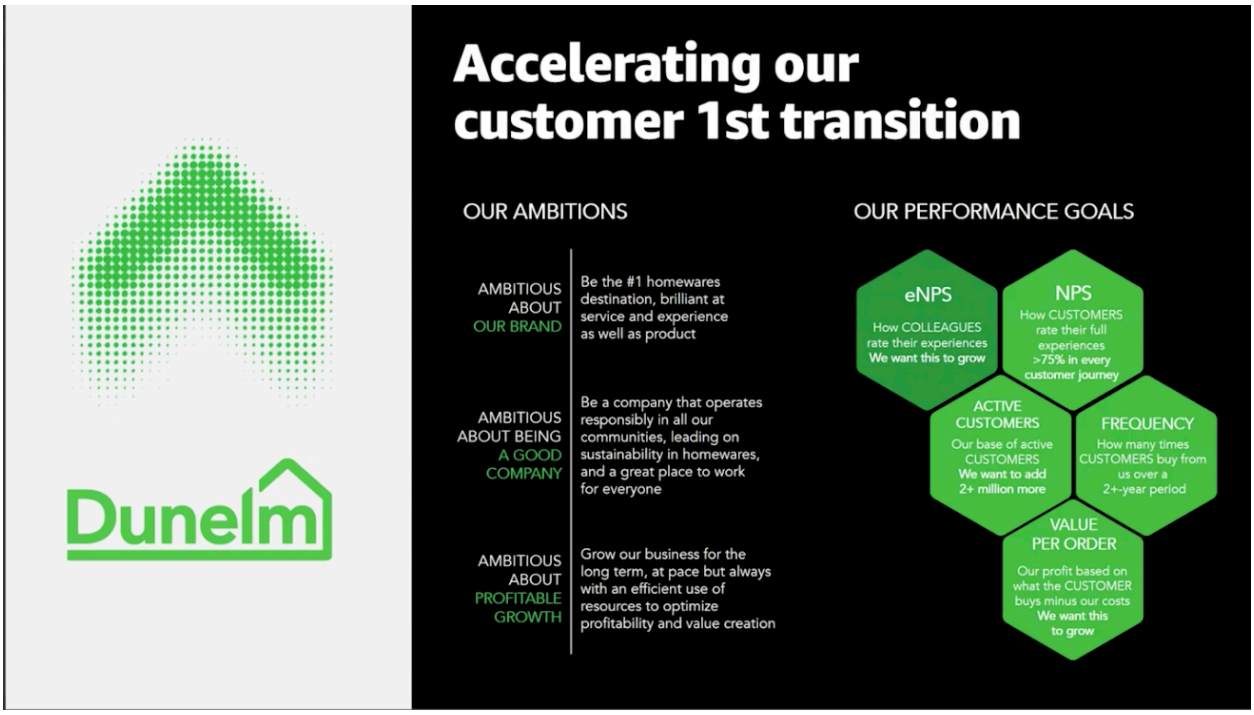


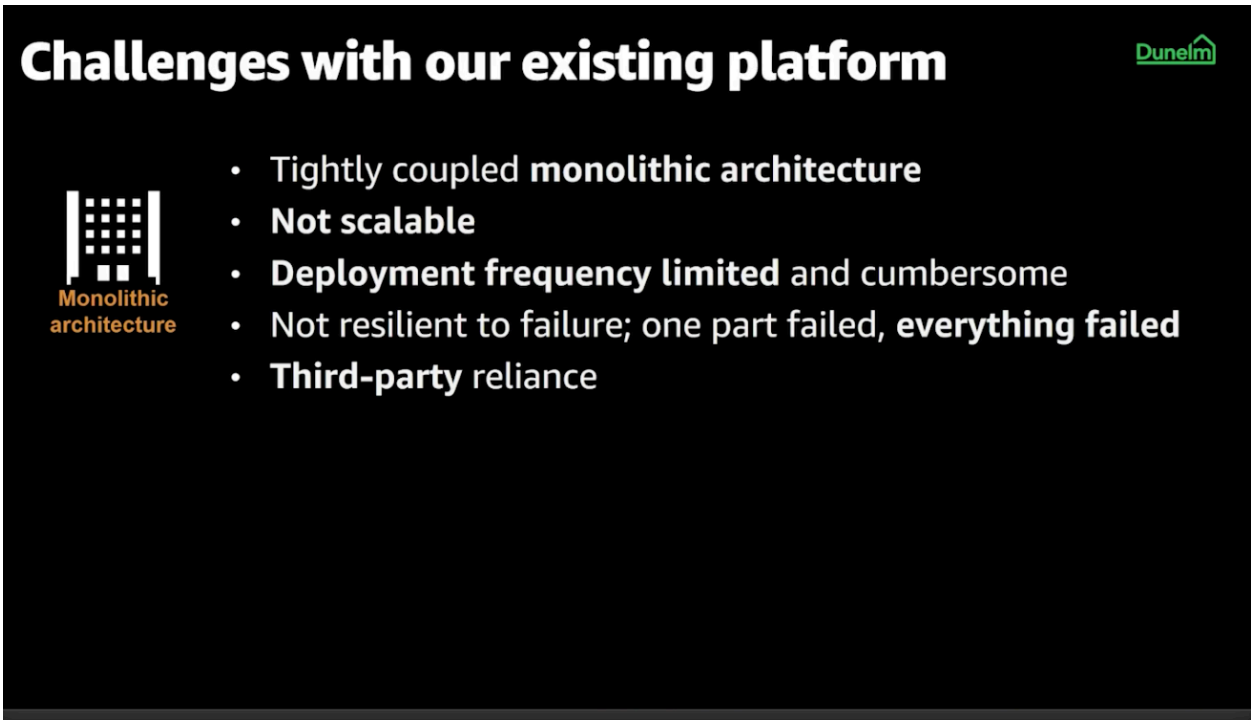
Write code to APIs instead of writing code to run in a specific place. Building blocks to fit together rather than abstractions.
- AWS Lambda (application platform as an API) => write, upload code, run on demand.
- AWS Fargate => containers on demand

Can call code via SQS, which is powerful!

Case study: Dunelm

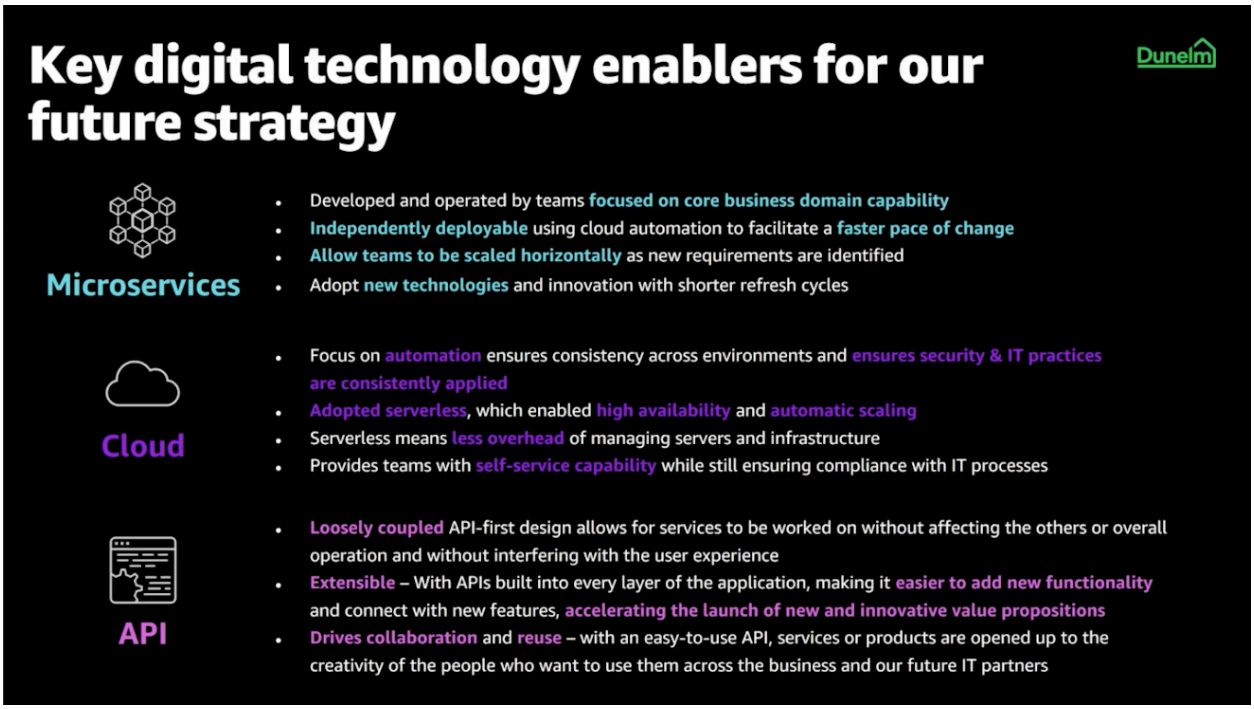






Long feedback loop, monthly deployments.

Decided to move to Serverless.

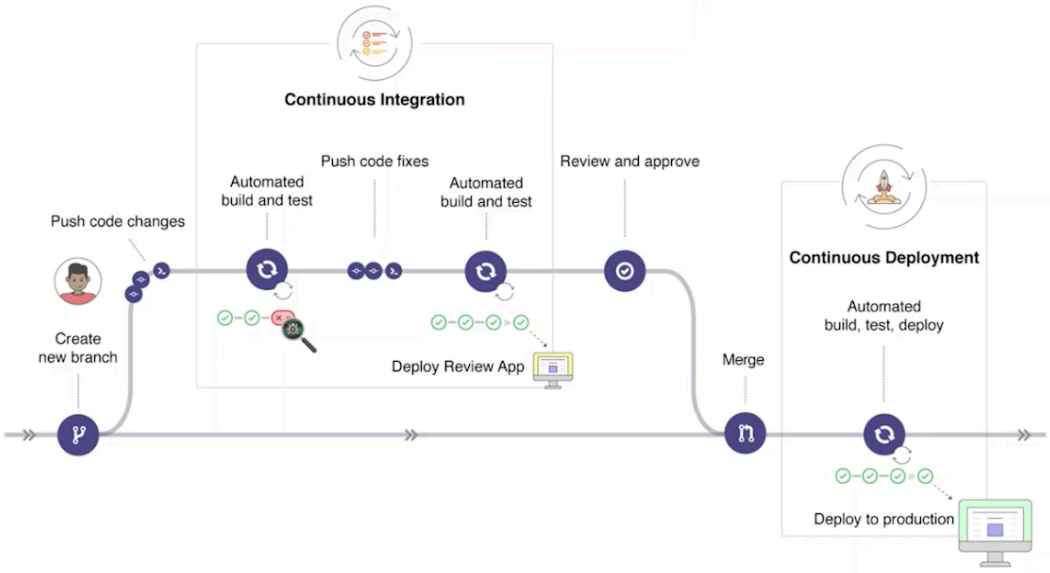


Scale faster, de-couple systems and services => autonomous.

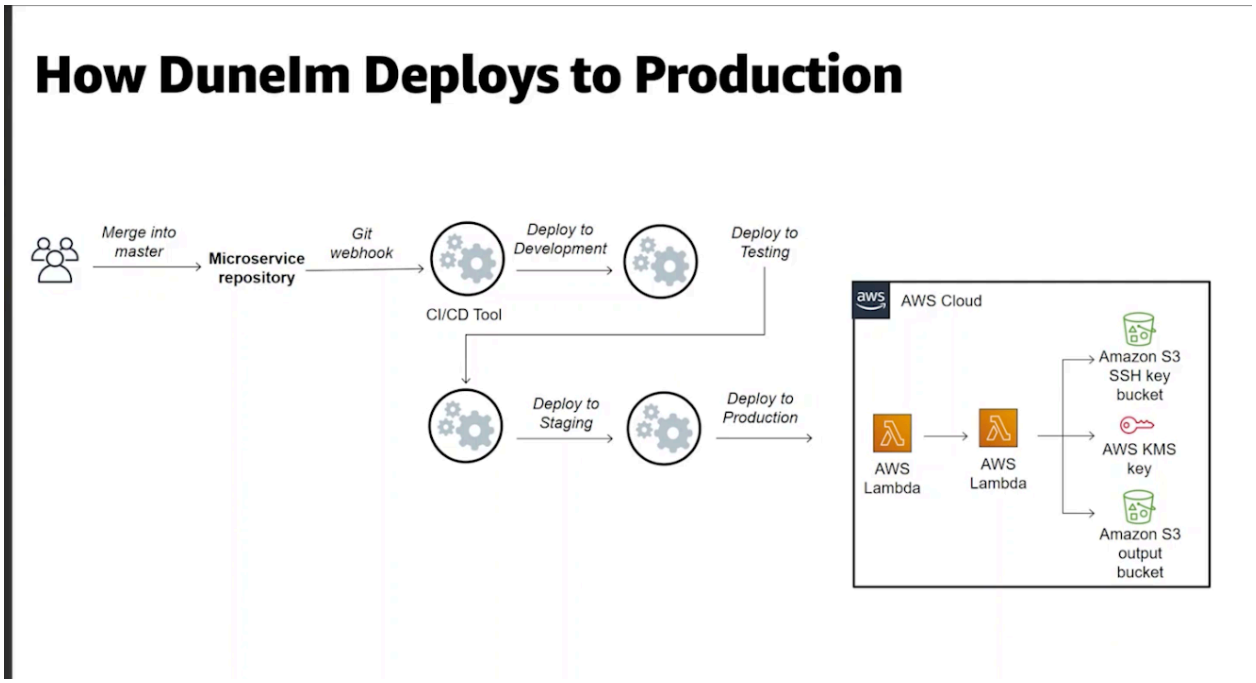
Moving from Component Teams to Outcome Teams

- Started with Component Teams who owned standalone services i.e. Search, Basket, and Payment. Got the services dev underway quickly, ensured standards and service ownership
- Does however create dependencies between the teams. We managed this but it did create an overhead on coordination and e2e testing
- Now well into our journey shaping Outcome Teams, giving them autonomy and ownership to drive the strategic outcomes
    - Less about giving the work to do, more about understanding the problem and letting the teams respond through their own innovation and creativity
    - Equipping our teams with the expertise they need to make changes to all services required for their outcome
    - Needs a cultural mind shift , teams need to learn other systems and push beyond their comfort zone
    - Outcome Examples - "Reduce basket abandonment by 5%" or "Reduce deployment lead times"

How DuneIm Creates New Microservices



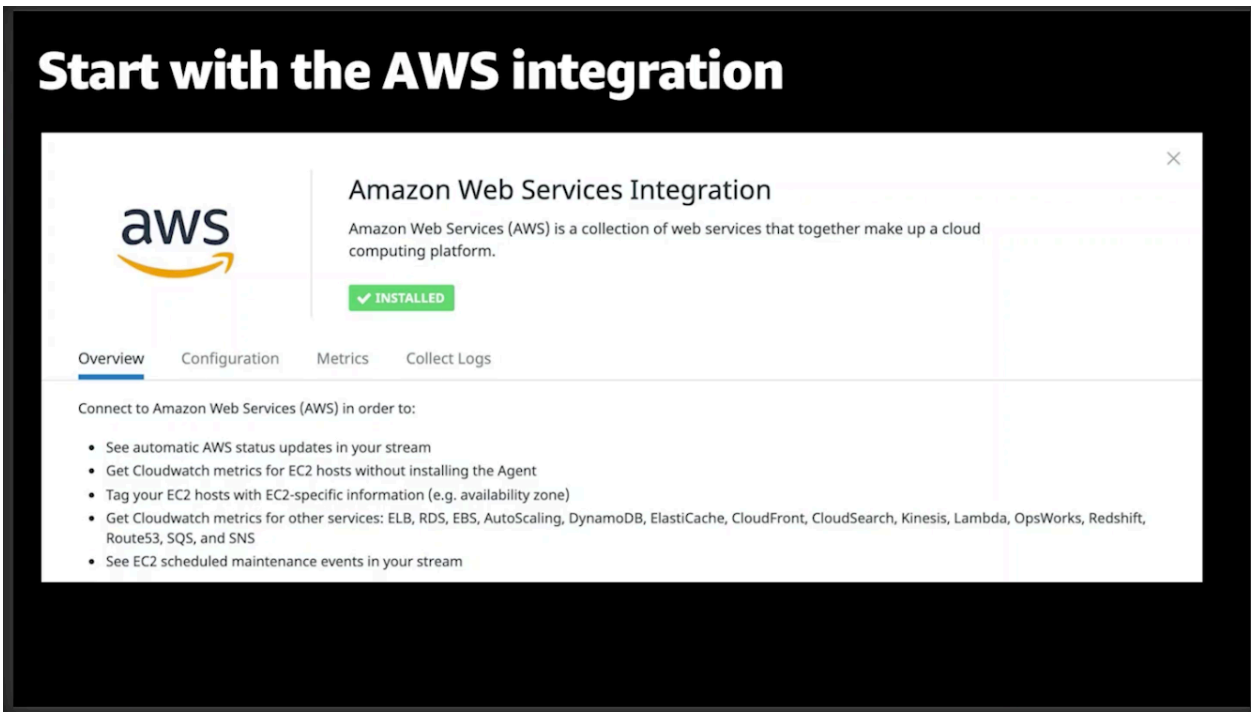
This gives teams more autonomy.

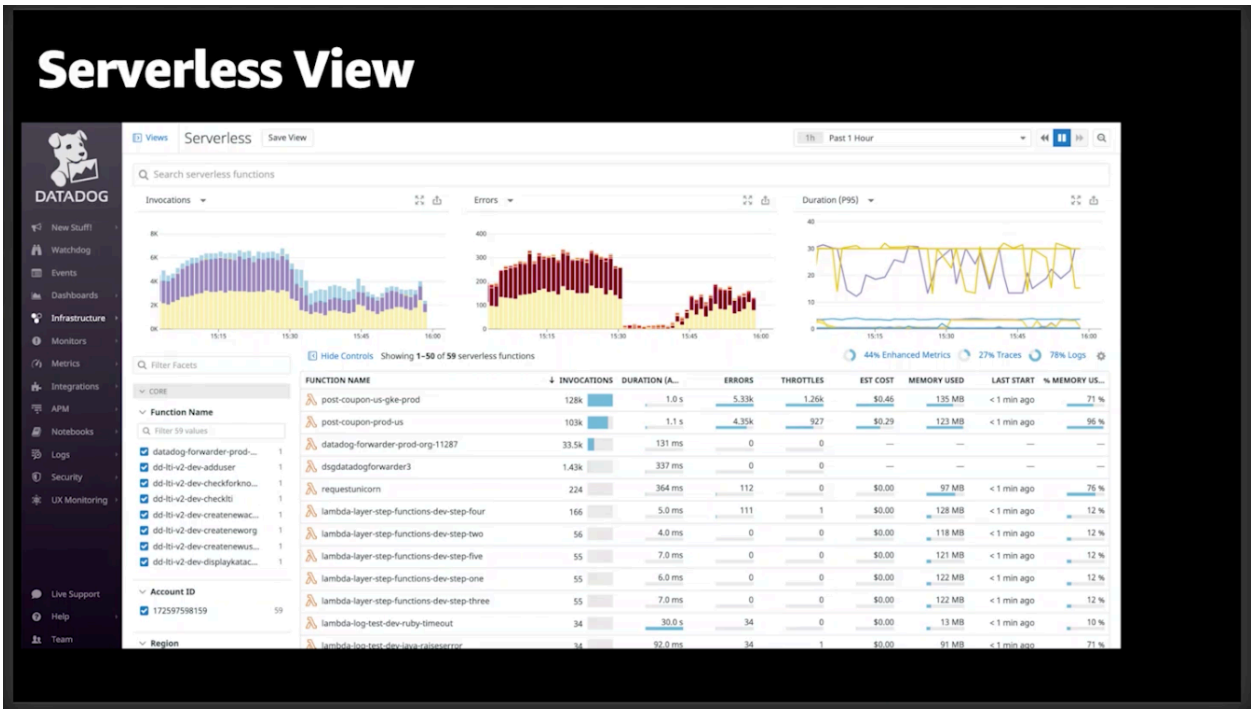


Deployment flow.

Datadog Pitch:

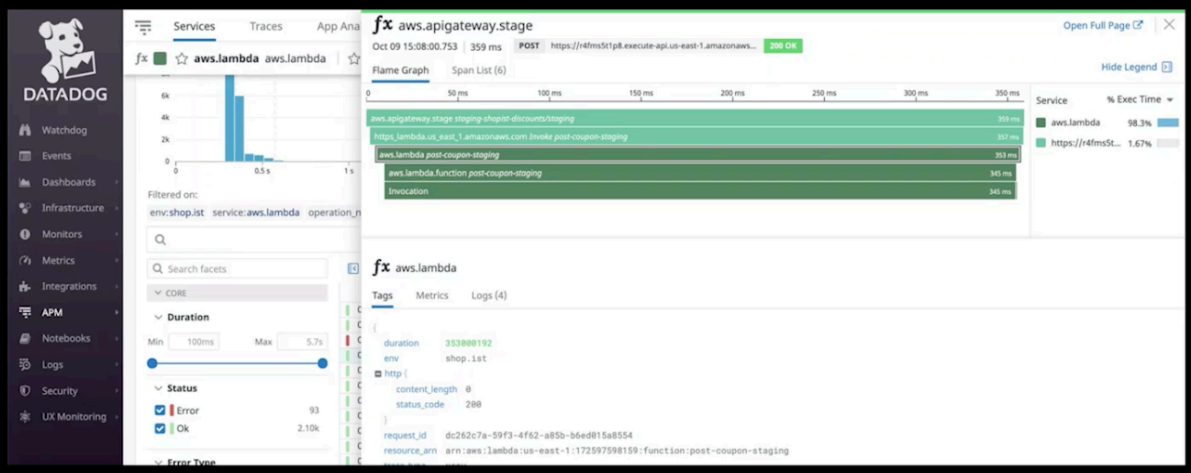
Tradeoffs of switching to Serverless: Where do you begin when debugging? What if multiple services are involved? No servers to look at anymore. If many people deploying at the same time, how to determine who pushed the breaking change?



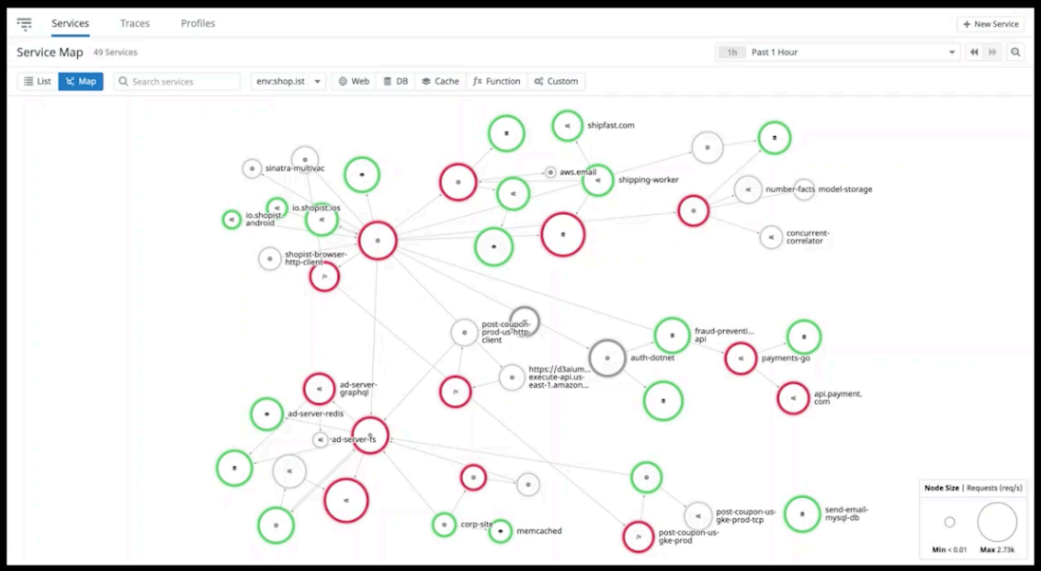


No need to communicate to team that you are about to deploy: everyone has shared view.

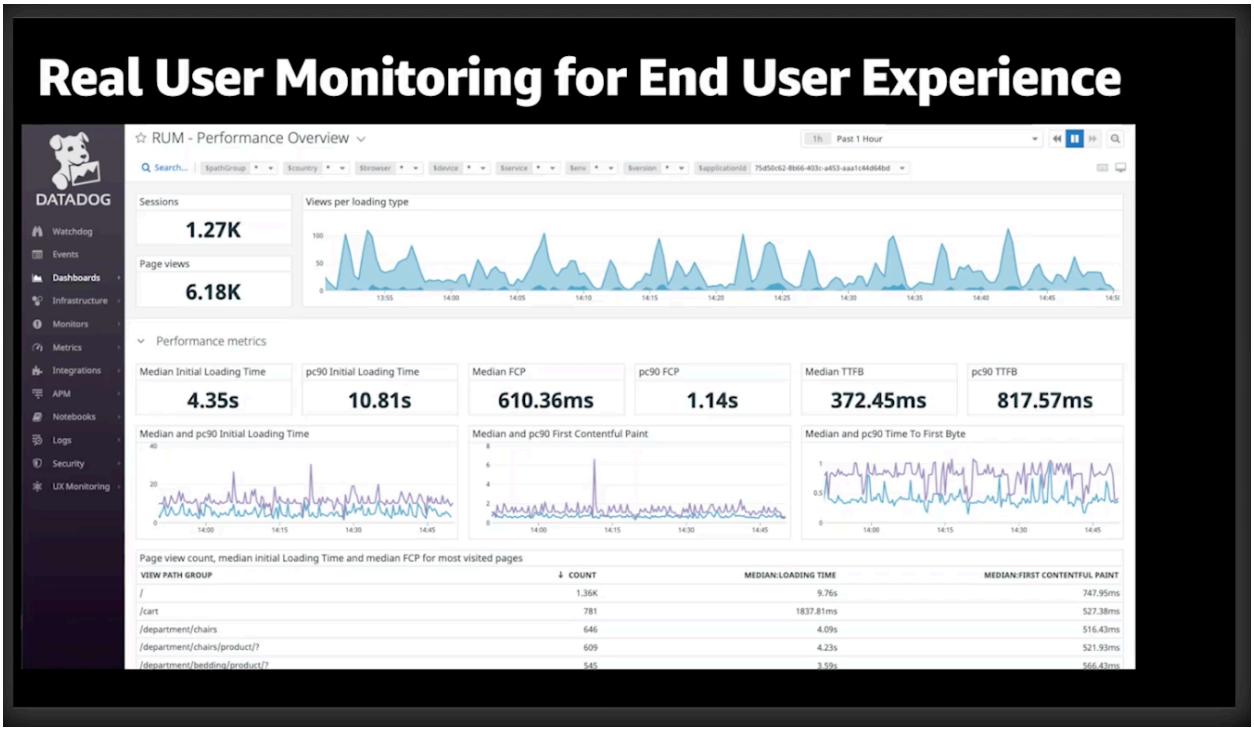
Add APM and logs for application-level insights with Datadog Forwarder



Service Map for Architecture Changes Over Time

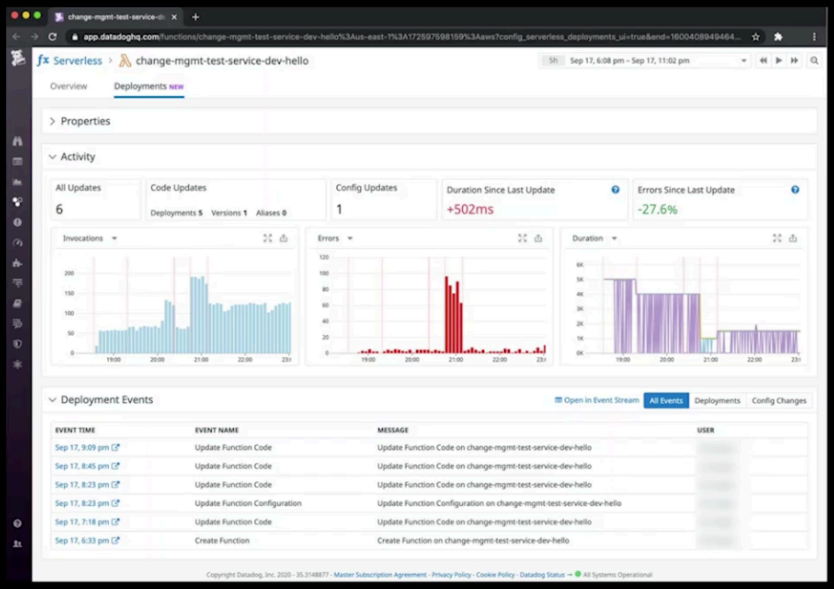


-05:3



Another challenge with Serverless: How to track deployments over time? Track AWS Lambda versions over time.

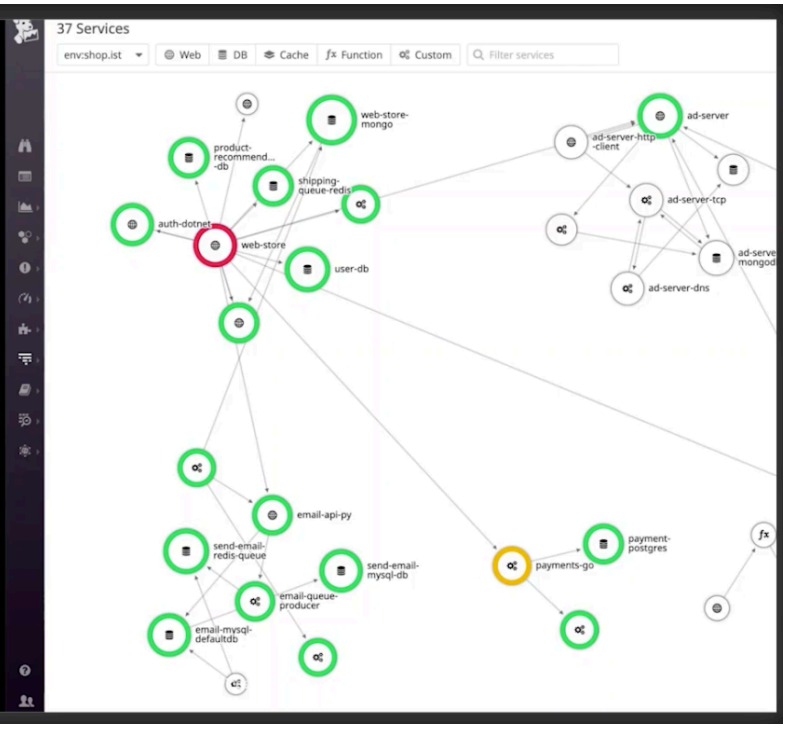
Track AWS Lambda versions over time



Zero Instrumentation Serverless

With AWS SAM and CDK Integrations

Zero instrumentation serverless observability with Datadog's AWS SAM Macro



Alerts as Code with AWS SAM

```
Resources:
  [...]
  MyTestMonitor:
    Type: Datadog::Monitors::Monitor
    Properties:
      Type: metric alert
      Query: 'avg(last_5m):max:aws.lambda.enhanced.duration{*} by {functionname} /
(avg:aws.lambda.timeout{*} by {functionname} / 80 ) > 0.9'
      Name: "Function duration is 90% of limit, increase timeout in AWS Console."
      Multi: true
      Options:
        Thresholds:
          Critical: 0.9
          Warning: 0.7
      DatadogCredentials:
        ApiURL: https://api.datadoghq.com
        ApiKey: <DATADOG_API_KEY>
        ApplicationKey: <DATADOG_APP_KEY>
```