Visualizing Serverless Architectures

What makes a healthy serverless application?

About Me

- @nodebotanist
- Developer Relations Engineer at IOpipe
- Hardware maker/student/ author
- I <3 Serverless and IoT!

The state of serverless ops

- Firmly in the reactionary stage in most cases
  - "Oh no, it's down, why is it down?!"
- We keep some metrics, but there's so many unknowns!

The ideal: Preventative Operations

- Being able to observe your serverless application from the 10,000m view down to each invocation
- Being able to see metrics over time, to see problems before they arise
- Instead of wondering why the app is down, prevent the problems that would take it down

But how do we get from reactionary to preventative?

The Observability Craze

- Monitoring is out, observability is in
- Observability allows us to handle unknowns better in this chaotic, uncharted new space
- It's less of a craze and more of an evolution in operations technique

HOWEVER,

METRICS ARE STILL IMPORTANT!

Metrics are still important

- We need some landmarks from which to gauge where we're going and where we've been
- It's hard to ingest large amounts of observed data without categorization
- Metrics, when planned well, are easy to turn into easily-digestable, usable visualizations

Some common serverless metrics

Memory usage/CPU Usage

- Spot memory leaks over time
- Know when to scale up (or down!) your serverless functions
- Know when to split off a task or use an external service

Errors

- You want to know when, why, and how a function invocation fails
- You want to know if there's a ripple, and where that ripple begins
- Is the cause your app, or a dependency being down, or a provider outage?
- What is the user seeing when your function errors?

Error Rate

- You don't need to sound the alarms (necessairily) if 100 invocations cause an error...over 1,000,000 invocations total (.01% error rate)
- The rate of errors to invocations is a key piece of perspective when evaluating the health of a serverless application!

Cold Starts

- If you have a high rate of cold starts, you'll have to factor in how this affects your performance
- You need to be able to tell why there is a high rate of cold starts: is the function not called often enough? Or are there peaks of activity that require lots of instances to be spun up all at once?

There are metrics out there
That pertain to your app

Custom Metrics

- One of the key benefits of observability is the ability to define your own custom metrics that fit your particular goals
- The amount of time a certain task takes, the user data that impacts your function, or anything you can think of!
- The downside: this does require instrumentation of your code

Demo Time: Observing an AWS Lambda application in IOpipe

The other key element: Time

- Data is most useful when aggregated and visualized over time
- Staring at the data during an incident only gives you the knowledge to fix it now, not necessarily prevent it from happening again
- Time equals context, and context creates lasting solutions!

Speaking of time…

The world of serverless observability and operations is trying to keep pace with an explosion of growth in serverless use. But the tooling is working its way through, and the future is bright!

Thanks for Listening!

- Kassandra Perch
- @nodebotanist
- Developer Relations Engineer at IOpipe