

The Ops in Serverless

Jennifer Davis

Senior Cloud Advocate, Microsoft

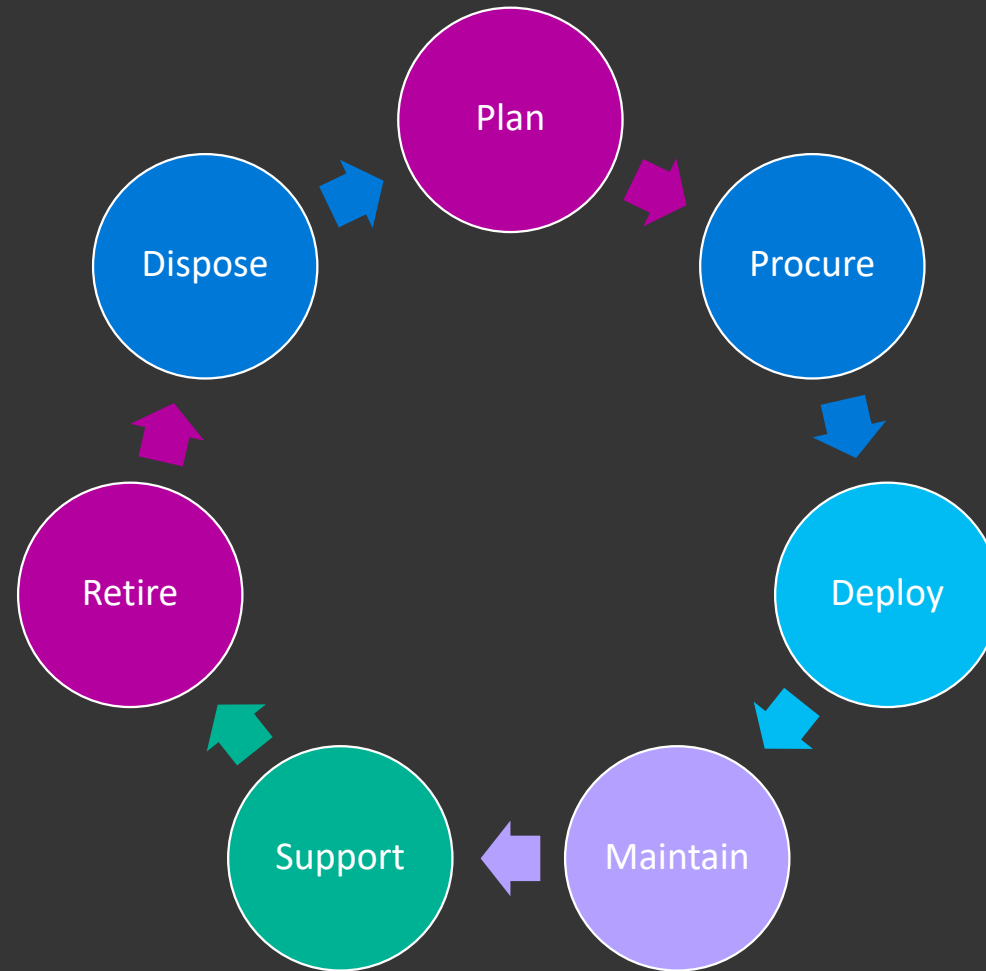


Defining Serverless (FaaS)

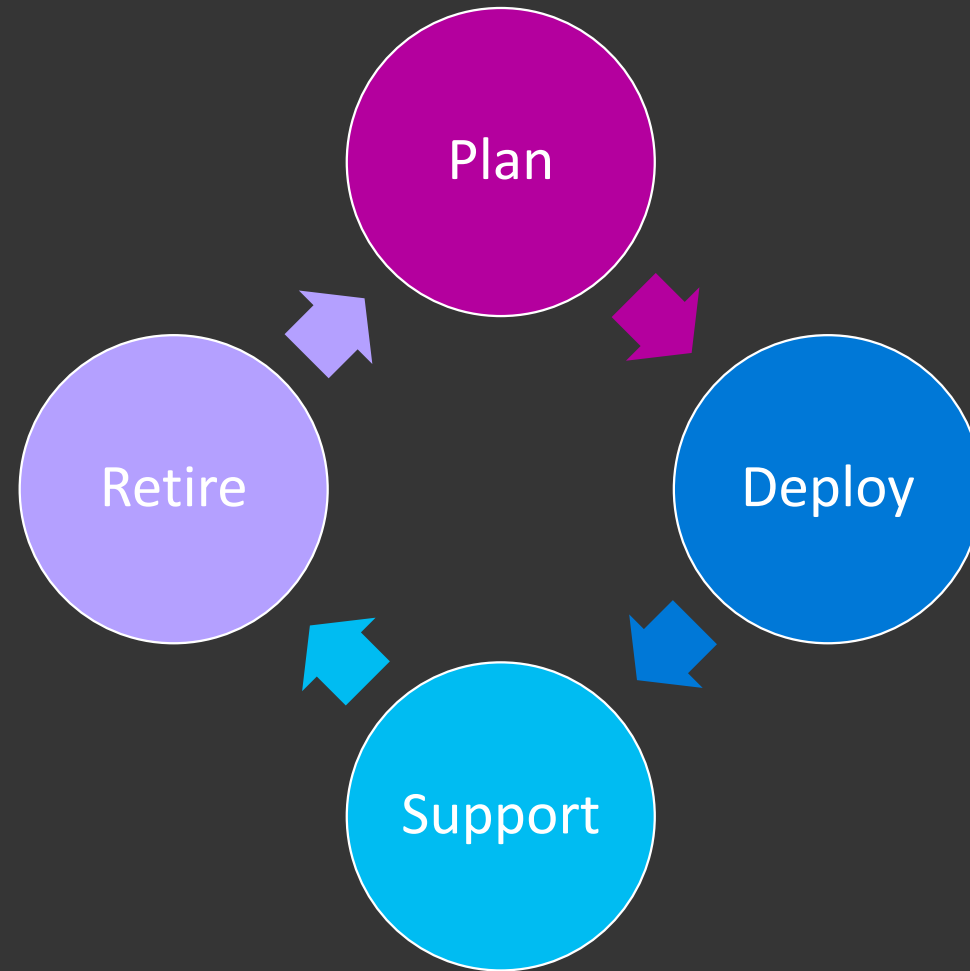
- No server provisioning
- Scales automatically
- No idling costs
- Increased availability



Compute Lifecycle



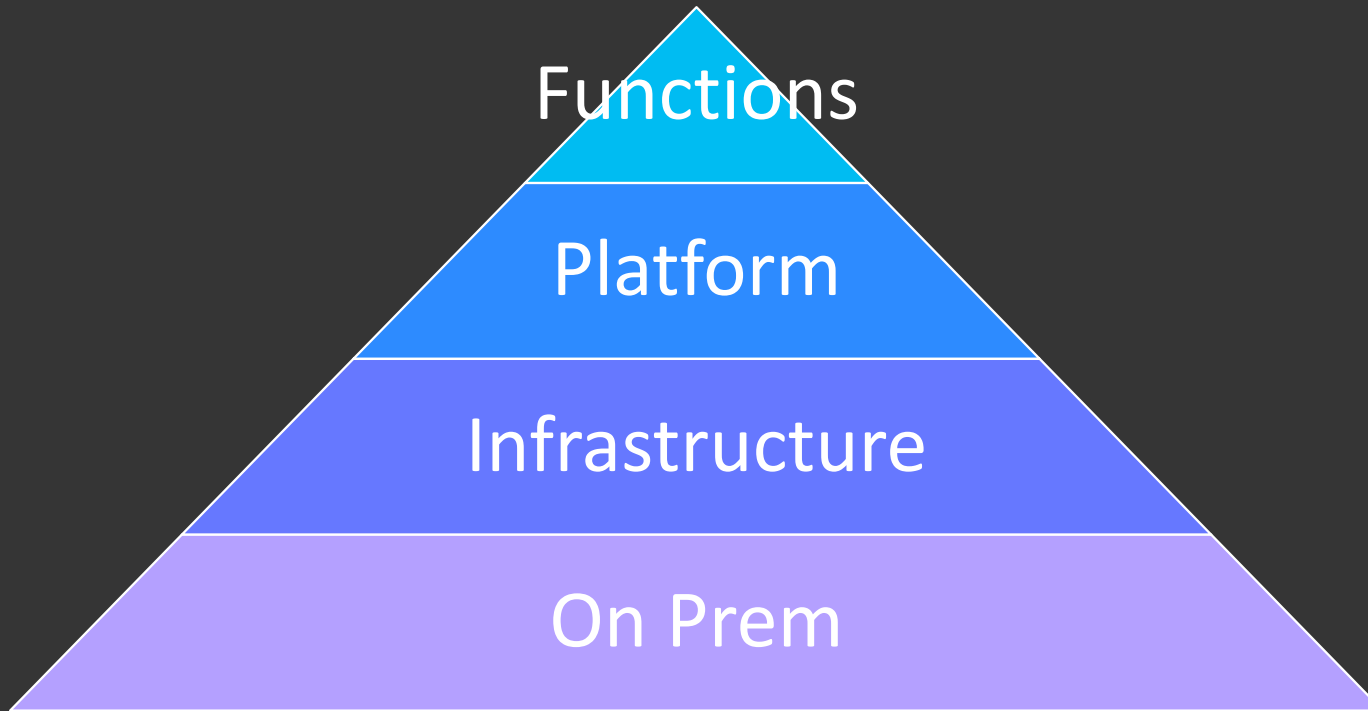
Serverless Lifecycle



Benefits of Serverless

Additional Choice

Reduction of cost




@sigje
#serverlessops



Governance



API Management policies

11/19/2017 • 4 minutes to read •  +2

This section provides a reference for the following API Management policies. For information on adding and configuring policies, see [Policies in API Management](#).

Policies are a powerful capability of the system that allow the publisher to change the behavior of the API through configuration. Policies are a collection of Statements that are executed sequentially on the request or response of an API. Popular Statements include format conversion from XML to JSON and call rate limiting to restrict the amount of incoming calls from a developer. Many more policies are available out of the box.

Policy expressions can be used as attribute values or text values in any of the API Management policies, unless the policy specifies otherwise. Some policies such as the [Control flow](#) and [Set variable](#) policies are based on policy expressions. For more information, see [Advanced policies](#) and [Policy expressions](#).

Policies

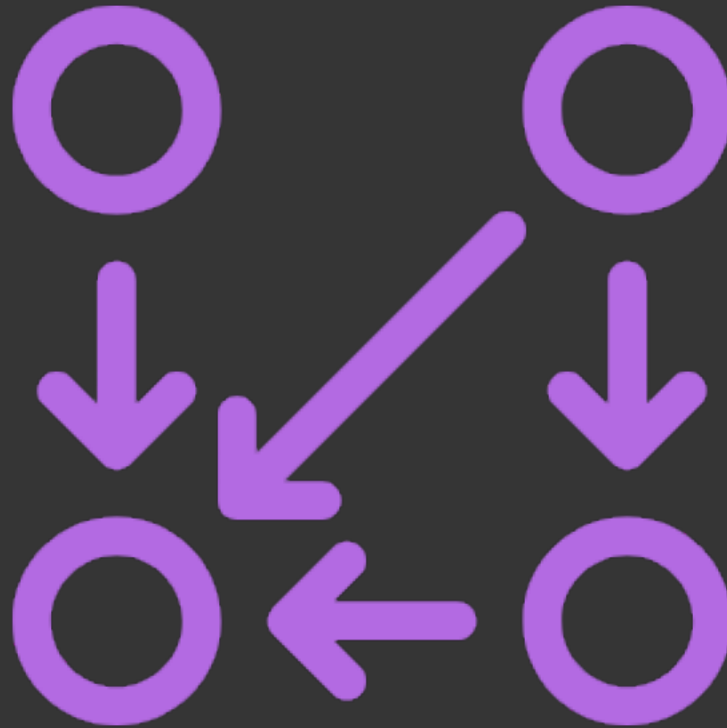
- [Access restriction policies](#)
 - [Check HTTP header](#) - Enforces existence and/or value of a HTTP Header.
 - [Limit call rate by subscription](#) - Prevents API usage spikes by limiting call rate, on a per subscription basis.
 - [Limit call rate by key](#) - Prevents API usage spikes by limiting call rate, on a per key basis.
 - [Restrict caller IPs](#) - Filters (allows/denies) calls from specific IP addresses and/or address ranges.
 - [Set usage quota by subscription](#) - Allows you to enforce a renewable or lifetime call volume and/or bandwidth quota, on a per subscription basis.
 - [Set usage quota by key](#) - Allows you to enforce a renewable or lifetime call volume and/or bandwidth quota, on a per key basis.
 - [Validate JWT](#) - Enforces existence and validity of a JWT extracted from either a specified HTTP Header or a specified query parameter.
- [Advanced policies](#)
 - [Control flow](#) - Conditionally applies policy statements based on the evaluation of Boolean expressions.
 - [Forward request](#) - Forwards the request to the backend service.
 - [Limit concurrency](#) - Prevents enclosed policies from executing by more than the specified number of requests at a time.
 - [Log to Event Hub](#) - Sends messages in the specified format to a message target defined by a Logger entity.
 - [Mock response](#) - Aborts pipeline execution and returns a mocked response directly to the caller.
 - [Retry](#) - Retries execution of the enclosed policy statements, if and until the condition is met. Execution will repeat at the specified time intervals and up to the specified retry count.
 - [Return response](#) - Aborts pipeline execution and returns the specified response directly to the caller.
 - [Send one way request](#) - Sends a request to the specified URL without waiting for a response.
 - [Send request](#) - Sends a request to the specified URL.
 - [Set HTTP proxy](#) - Allows you to route forwarded requests via an HTTP proxy.

bit.ly/AzureAPIPolicies



@sigje
#serverlessops

Dependencies



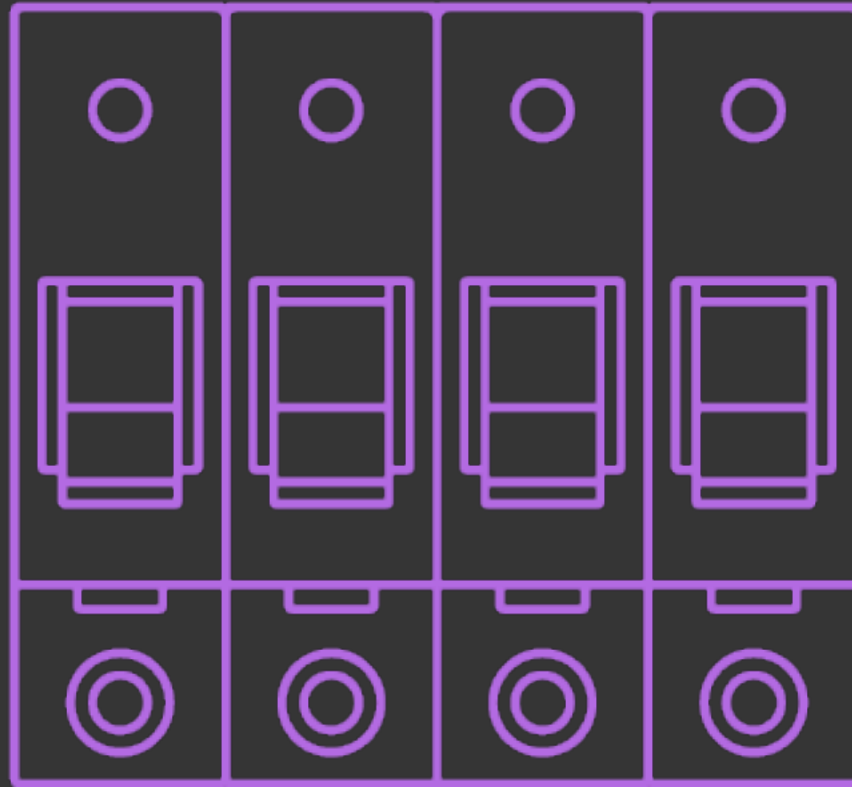
@sigje
#serverlessops

Function Rot



@sigje
#serverlessops

Capacity Planning



@sigje
#serverlessops

Service Resilience

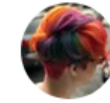


@sigje
#serverlessops

Monitoring SLI/SLOs

System Metrics

- CPU
- Memory
- Disk Utilization



systemd.timer mom
@sophaskins

Following



a graph that tells a story:

CPU usage of a tool that normally uses almost none



10:10 AM - 18 Oct 2018

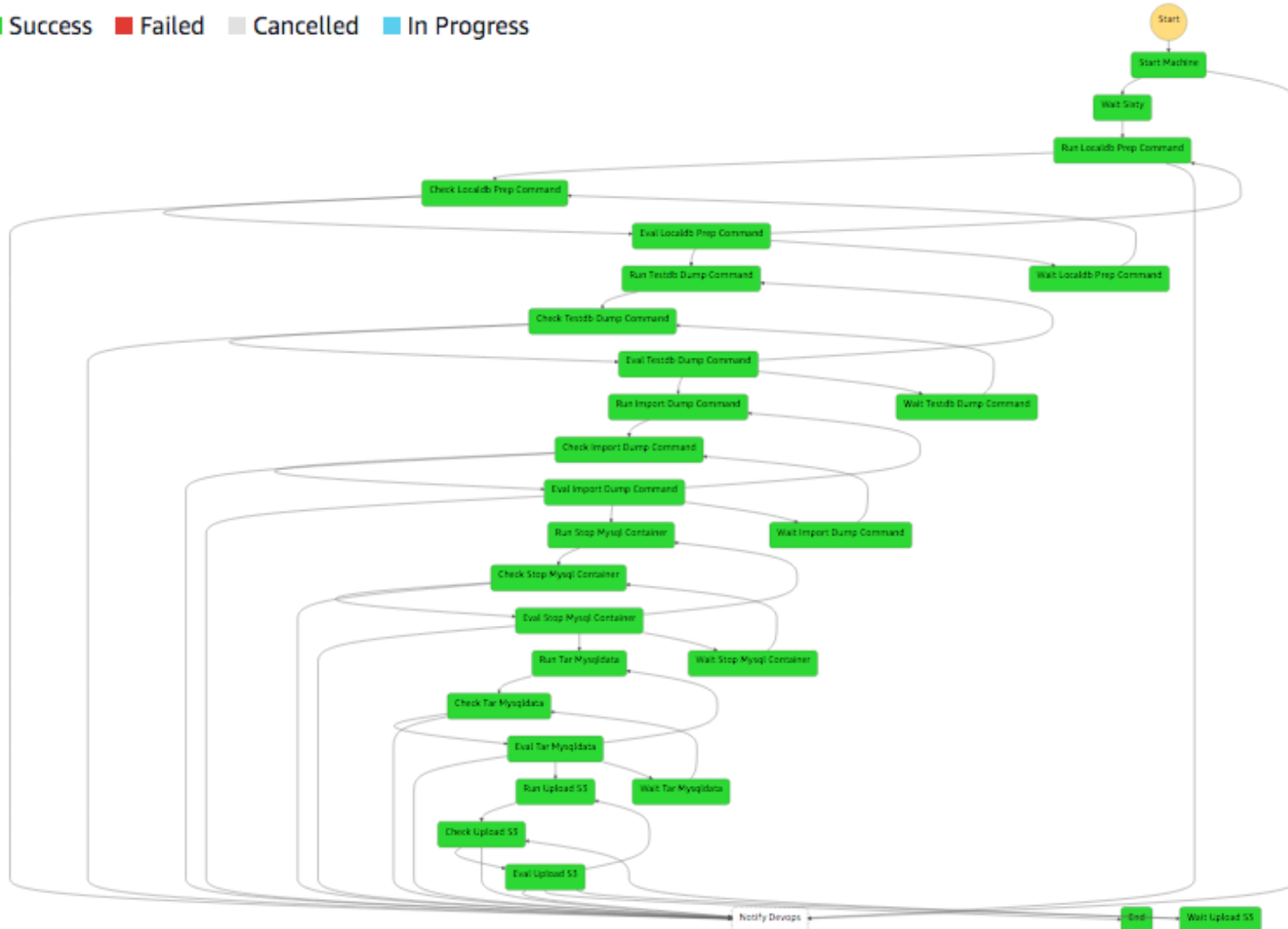


@sigje
#serverlessops

AWS Stepfunctions Metrics

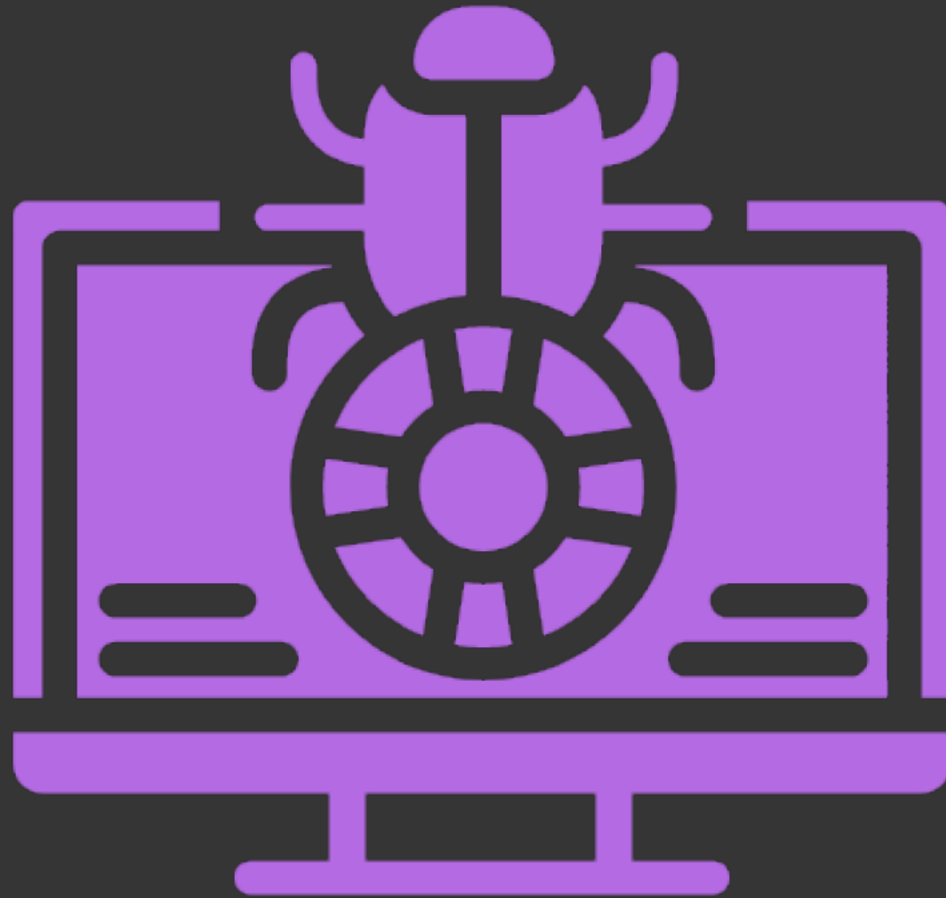
Visual workflow

■ Success ■ Failed ■ Cancelled ■ In Progress



- Duration
- Invocations
- Errors
- Throttles

Debugging

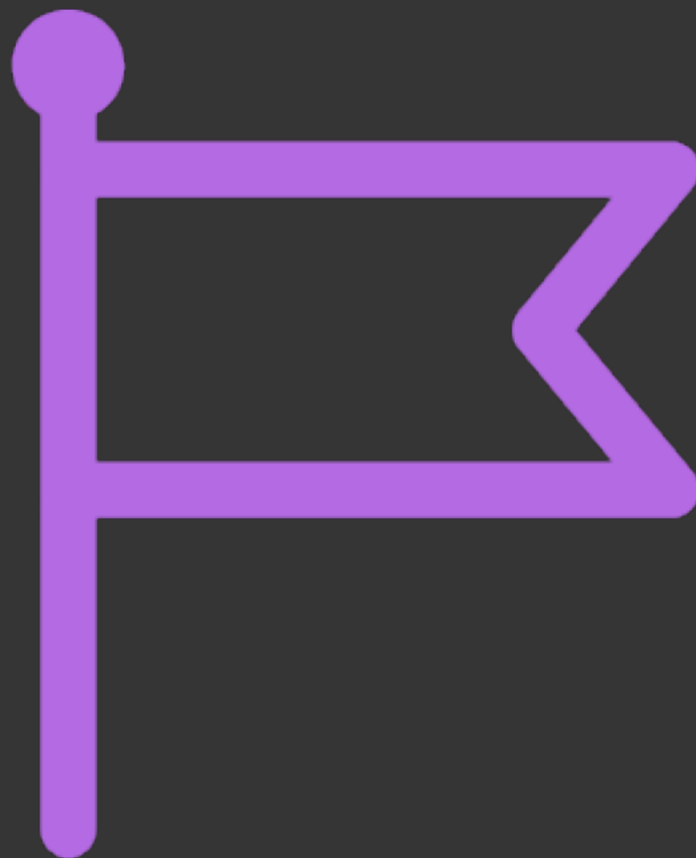


@sigje
#serverlessops

Instrumentation



Feature Flags/Toggles



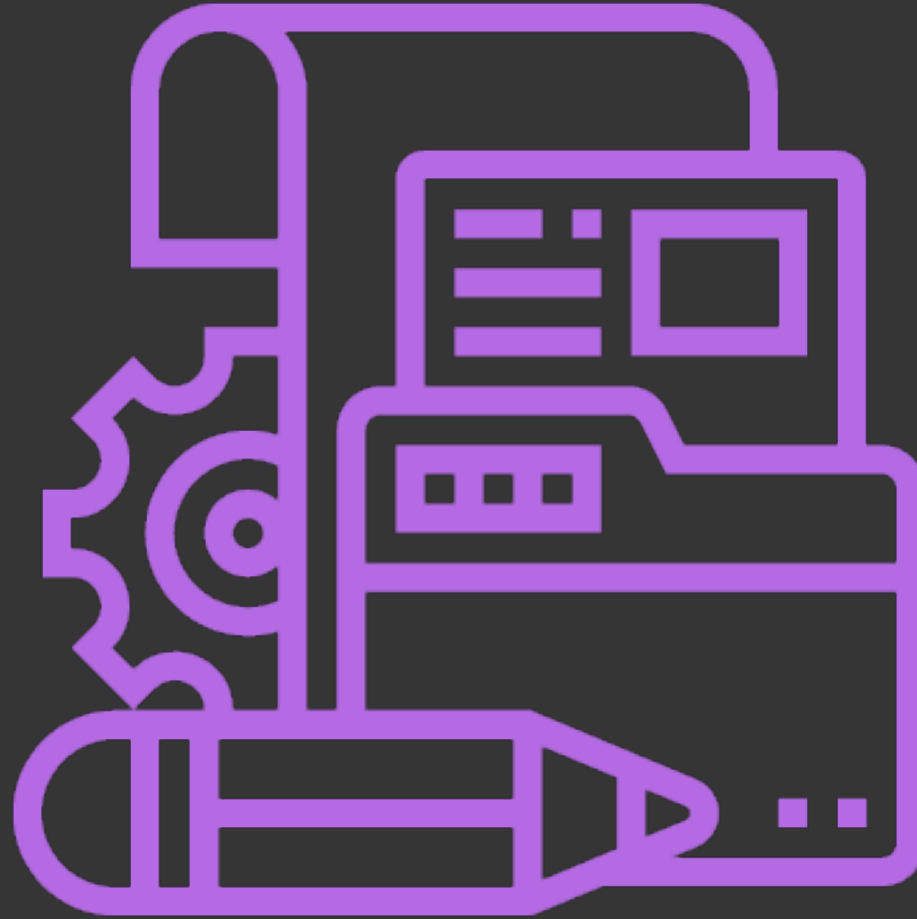
@sigje
#serverlessops

Testing



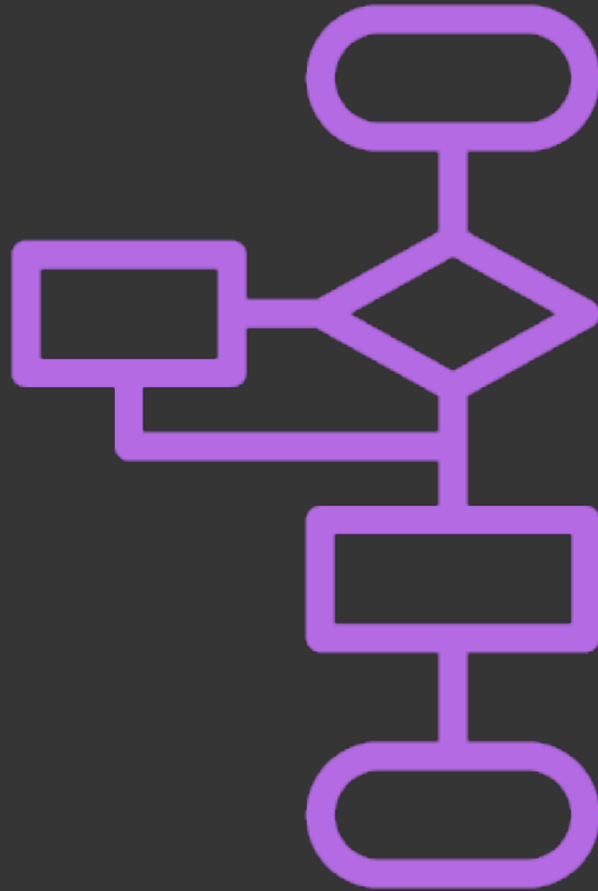
@sigje
#serverlessops

Documentation



@sigje
#serverlessops

Processes



@sigje
#serverlessops

Code



@sigje
#serverlessops



Bryan Friedman

@bryanfriedman

Follow



2001: I need a VM
2007: I need a lot of VMs
2012: Just give me PaaS
2013: Whoa containers!
2014: I need a lot of containers
2015: Whoa Kubernetes!
2017: I need a lot of Kubernetes
2018: So, bare metal?
2019: Wait, no, I actually need serverless
2020: Yeah...I need all the things

2:31 PM - 9 Jul 2019

Source:

<https://twitter.com/bryanfriedman/status/1148721255857573888>



@sigje

#serverlessops

What next?

- Join Slack conversations
 - [#serverlessops](https://serverless-forum.slack.com)
- Blogging?
 - Add operability/visibility/observability
- Email me!
 - serverless@awesomedevops.org



@sigje
#serverlessops