Serverless Architecture

Mischa Spiegelmock

Co-Founder, FETBRIDGE

About Me

- From Berkeley, California
- Co-founder of JetBridge in SF/Kyiv/Wrocław
- Do computer shit
- Been doing computer shit a long time
- I’m a **developer**
- AWS Certified Solutions Architect

This Talk

- Ops from a developer's point of view
- Cloud-native application architecture
- Pros/cons of serverless applications
- Costs of Serverless

Ops

As a developer

What’s the ☁️?

Just somebody else's computers?

- Use economies of scale of cloud providers
- Worry about higher-level issues more related to business
- Increase developer “leverage” 💪
- Don’t solve already solved problems (e.g. deploying web applications)

Serverless

Mindset

Business Value

Functions As A Service

"FaaS"

AWS Lambda

FaaS

- Write a function
- Deploy it “somewhere”
- Invoke it “somehow”

Write A Function

How it works

Run

"Hello 🌍!"

```
1  exports.handler = (event, context, callback) => {
2      // Succeed with the string "Hello world!"
3      callback(null, 'Hello 🌍!');
4  };
```

Deploy It Somewhere

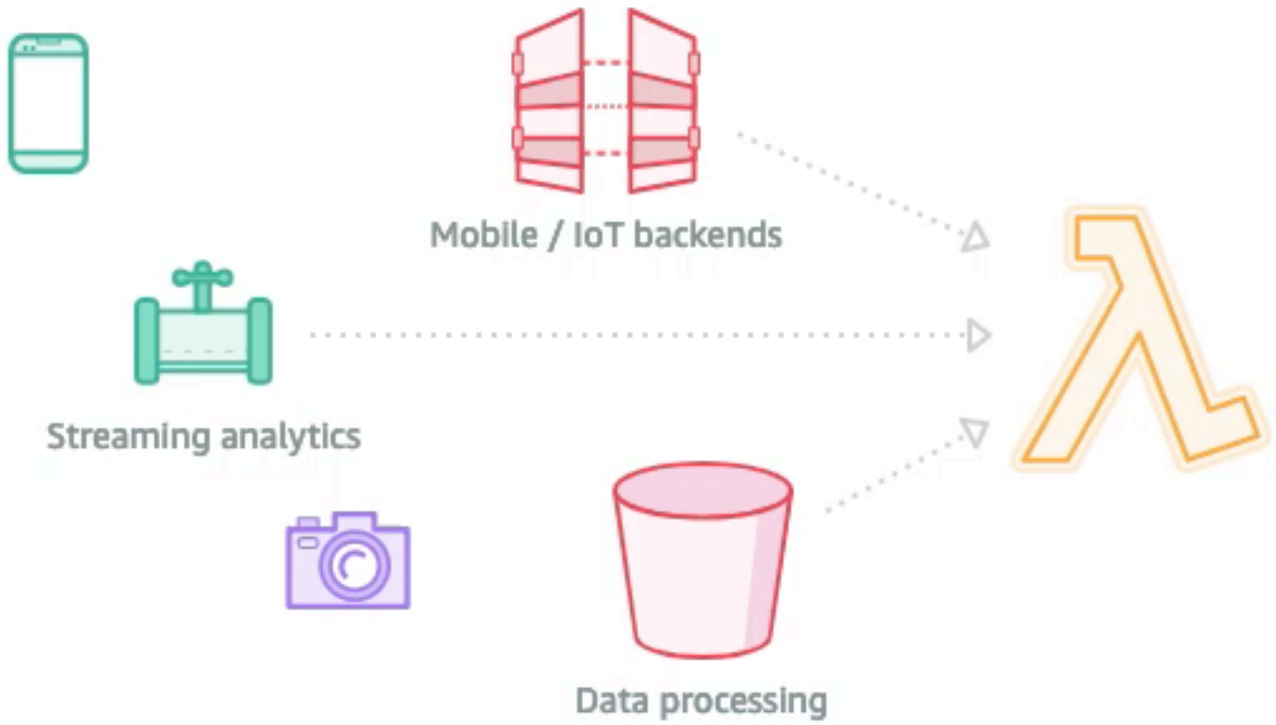
```
Last login: Mon Feb 25 17:12:46 on ttys015
➜  ~ slsp
```

Invoke It Somehow

How it works

Previous | Next: Scale seamlessly

```
6:49:04 PM: "Hello 🌍!"
```



Lambda responds to events

Once you create Lambda functions, you can configure them to respond to events from a variety of sources. Try sending a mobile notification, streaming data to Lambda, or placing a photo in an S3 bucket.

So what?

For Free:

High Availability

Not My Problem

Versioning

Every function gets a version

Autoscaling

1,000 polling concurrent invocations

Free-ish

Free as in 🍺

Function-Level Permissions

Least privilege resource access

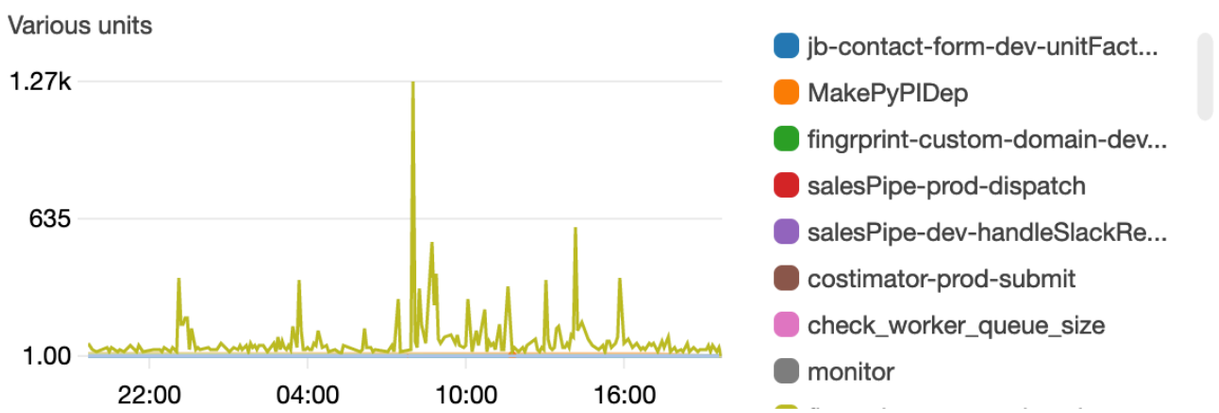
Expand all ◉ Row ○ Text

Filter events | all | 2019-02-24 (19:38:23)

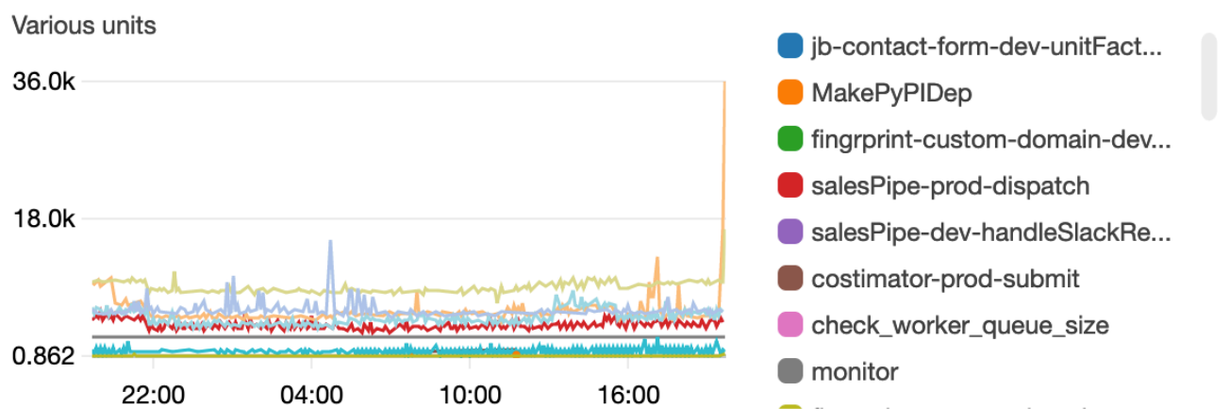
Time (UTC +00:00)	Message
2019-02-25	
19:24:34	START RequestId: 67c09671-9369-432e-9762-74e47662cfb9 Version: $LATEST
19:24:34	END RequestId: 67c09671-9369-432e-9762-74e47662cfb9
19:24:34	REPORT RequestId: 67c09671-9369-432e-9762-74e47662cfb9 Duration: 38.15 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 41 MB
19:24:34	START RequestId: bce0fba0-9390-4218-9a86-9557ba8f046c Version: $LATEST
19:24:34	END RequestId: bce0fba0-9390-4218-9a86-9557ba8f046c
19:24:34	REPORT RequestId: bce0fba0-9390-4218-9a86-9557ba8f046c Duration: 12.40 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 41 MB
19:24:34	START RequestId: 942d2d7b-71fe-4f86-be43-bc449b0d0bb3 Version: $LATEST
19:24:34	END RequestId: 942d2d7b-71fe-4f86-be43-bc449b0d0bb3
19:24:34	REPORT RequestId: 942d2d7b-71fe-4f86-be43-bc449b0d0bb3 Duration: 20.35 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 41 MB
19:24:42	START RequestId: d9d89820-fcd2-4178-bbc3-65dccde83dd5 Version: $LATEST
19:24:42	END RequestId: d9d89820-fcd2-4178-bbc3-65dccde83dd5

ALARM 0 INSUFFICIENT DATA 0 OK 0

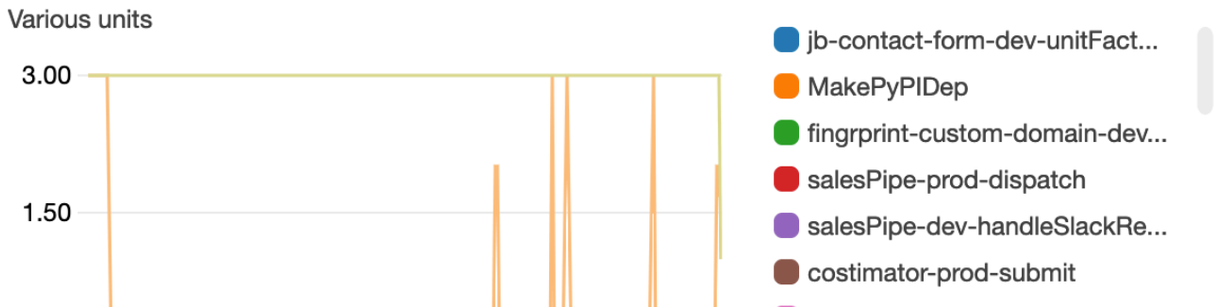
Invocations Sum



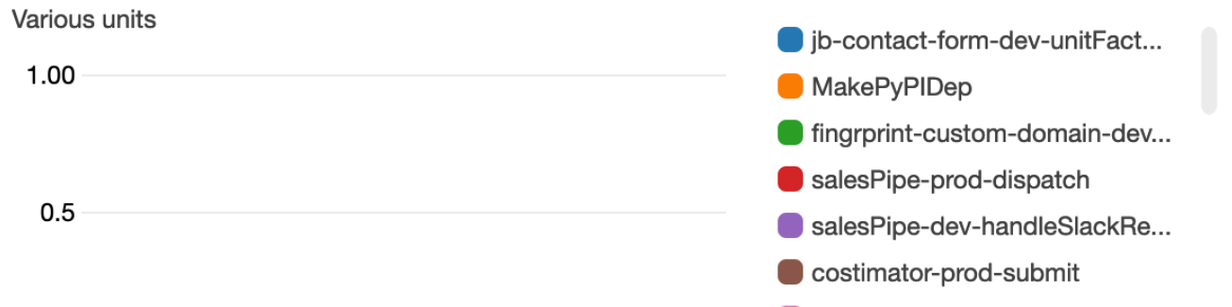
Duration Average



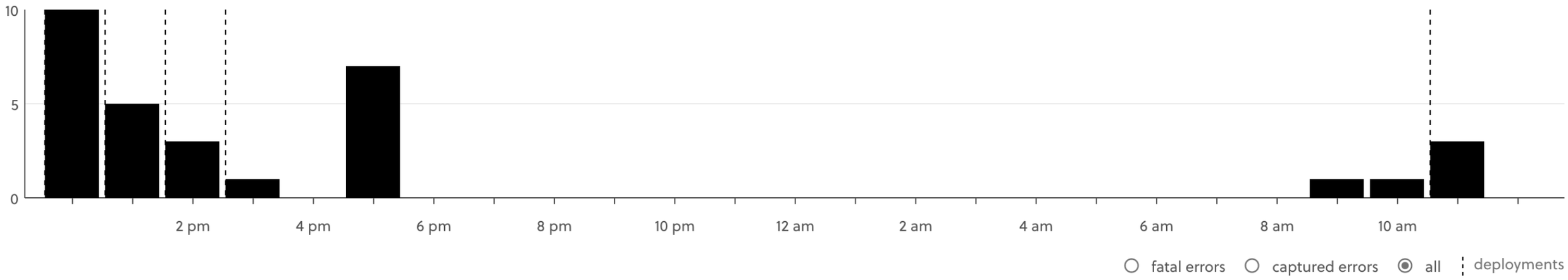
Errors Sum



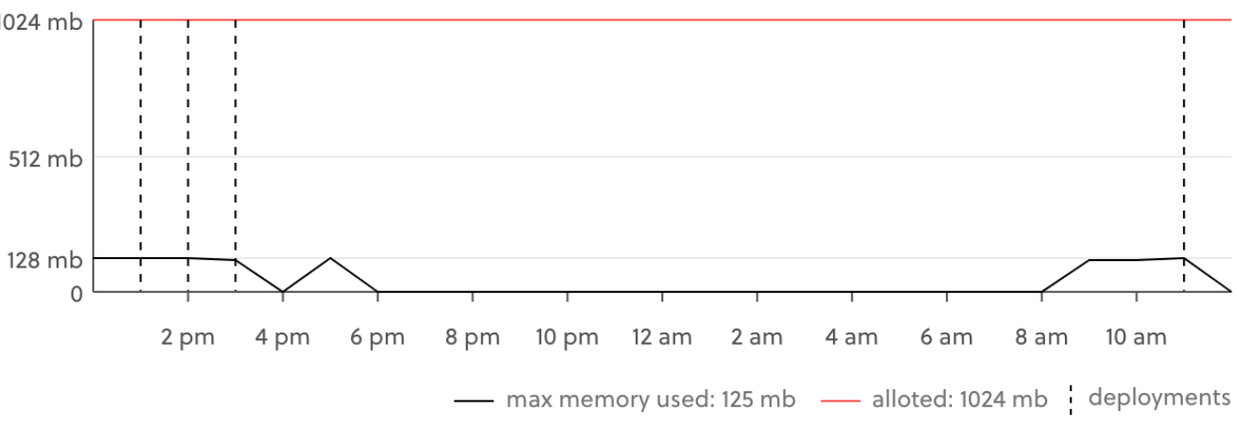
Throttles Sum



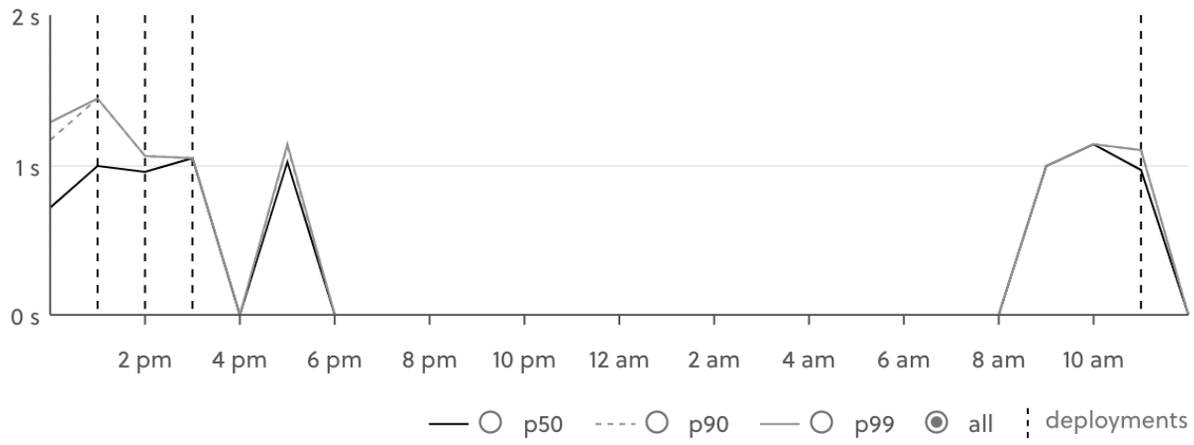
invocations & errors



memory usage



durations

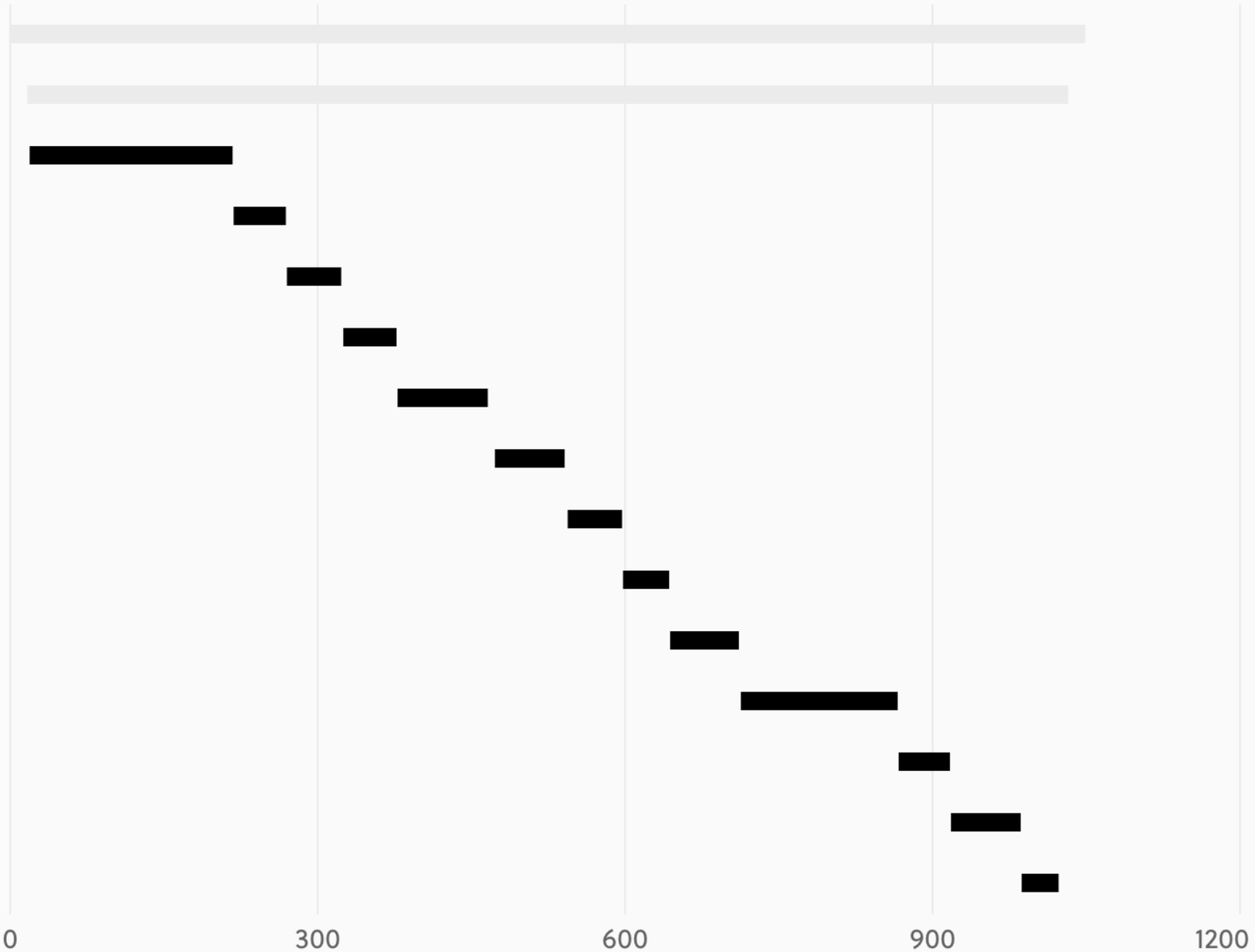


request

endpoint	/{proxy+}
method	POST
status	200
message	-
latency	1 s

spans

DURATION	OPERATION
1 s	total duration
1 s	function invocation
198 ms	CreateKeysAndCertificate iot
51 ms	CreateThing iot
53 ms	AttachPolicy iot
52 ms	AttachThingPrincipal iot
88 ms	CreateCoreDefinition greengrass
68 ms	DescribeStacks cloudformation
53 ms	GetFunctionDefinition greengrass
45 ms	GetSubscriptionDefinition greengr...
67 ms	GetLoggerDefinition greengrass
153 ms	CreateGroup greengrass
50 ms	AssociateRoleToGroup greengrass
68 ms	UpdateThingShadow iot-data
36 ms	DescribeEndpoint iot



view api requests | // × | /{proxy+} ×

all status codes | all methods | duration ms sec min 0 to 15

requests: 89

DATE & TIME	METHOD	ENDPOINT	DURATION	STATUS/MESSAGE
feb 12 11:35 am	POST	/{proxy+}	928	200: ok
feb 12 11:35 am	POST	/{proxy+}	2486	200: ok
feb 12 11:26 am	POST	/{proxy+}	2395	500: internal server error
feb 12 10:10 am	POST	/{proxy+}	2539	200: ok
feb 12 9:34 am	POST	/{proxy+}	2370	200: ok
feb 11 5:43 pm	POST	/{proxy+}	885	200: ok
feb 11 5:38 pm	POST	/{proxy+}	896	200: ok
feb 11 5:38 pm	POST	/{proxy+}	891	200: ok
feb 11 5:35 pm	POST	/{proxy+}	1040	200: ok
feb 11 5:34 pm	POST	/{proxy+}	2511	200: ok

general statistics

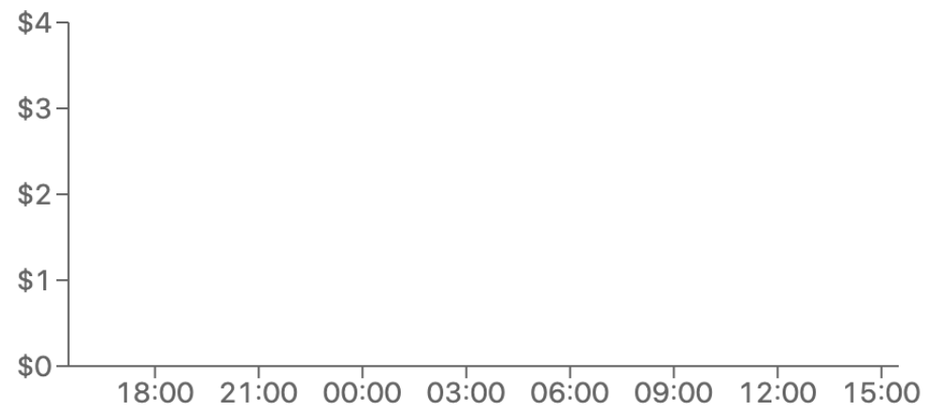
total invocations
8k

total errors
377

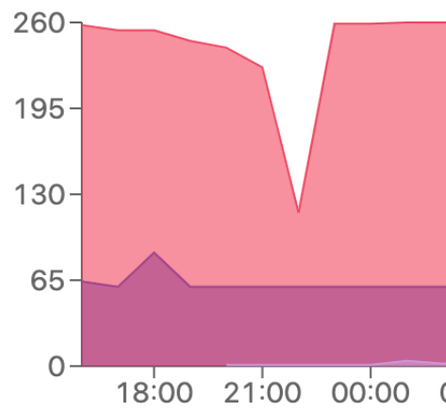
cost total
$ 0.05

billed duration
394 min

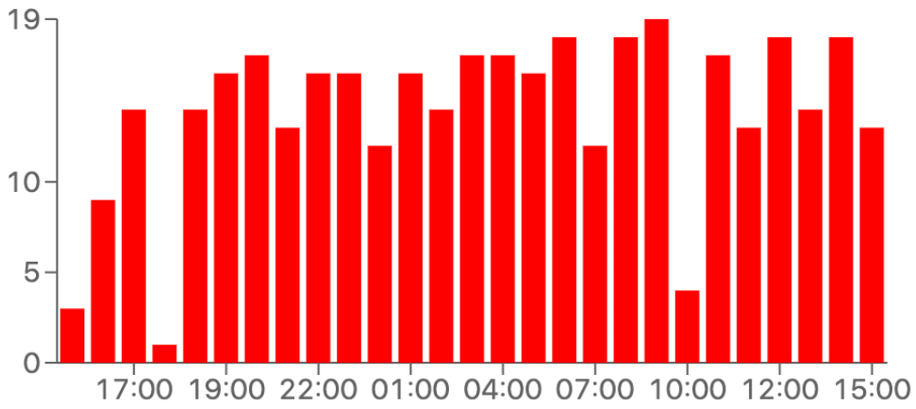
cost



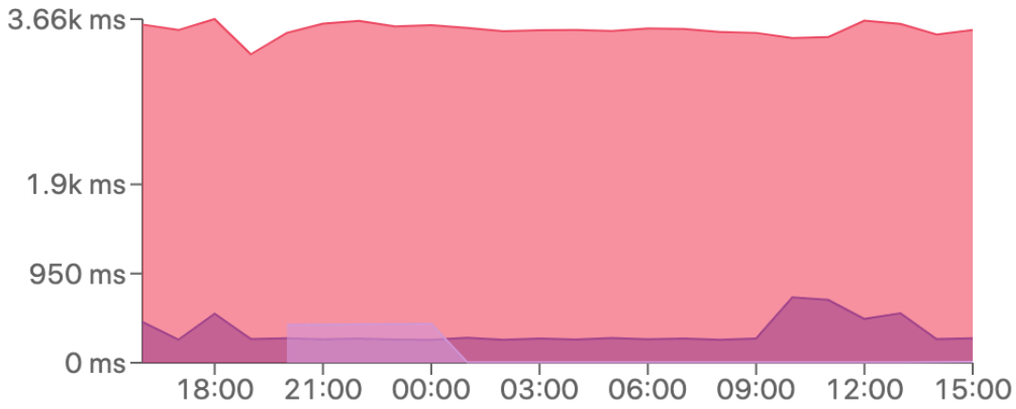
total invocations



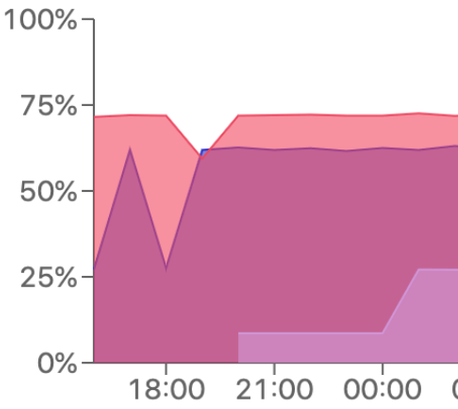
invocation errors



average durations



memory utilization



Building Blocks

- AWS Lambda
- Google Cloud Functions
- Kubeless
- Azure
- IBM

serverless framework

The easy and open way to build serverless applications

aws Azure Google Cloud Platform IBM Cloud kubernetes

serverless inc.

Serverless (AWS)

- Interface to CloudWatch Logs (logging)
- Interface to CloudFormation (IaC, deployments)
- Interface to API Gateway (HTTP)
- Interface to Parameter Store/Secrets Manager (secrets)
- Plugins

Additional Building Blocks

- API Gateway (HTTP)
- CloudFront (CDN, TLS)
- RDS, Aurora Serverless, DynamoDB, AppSync (SQL, NoSQL, GraphQL)
- Cognito (Authentication)
- IoT (Greengrass)
- SNS/SES (SMS/Email)
- Secrets Manager
- …

Benefits To FaaS

- Deployments: solved
- Infrastructure packaged with application
- Autoscaling
- High availability
- Tight integration with other building blocks
- No DevOps, just Dev

Downsides To FaaS

- Vendor lock-in
- Raw power
- Shared hosting
- Local development environment
- Cold start time

FaaS Use Cases

- Microservices
- Triggers (IoT, Storage, Message Queue, Auth, Alarms..)
- Web Applications
- Webhook Processors
- Streams (WebSockets, DynamoDB, Kinesis)

FaaS Is Not For

- Raw compute power (CPU, RAM, GPU, Disk)
- Long-running tasks
- Streaming media
- Large (disk size) applications
- High startup time
- Non-linux/amd64
- Dedicated hosting requirements (compliance)
- Real-time
- Probably a lot of other scenarios

Something In The Middle?

AWS Fargate

(managed containers)

Costs

GB/s

+

Invocations

	Price
Requests	$0.20 per 1M requests
Duration	$0.000016667 for every GB-second

Memory (MB)	Price per 100ms
128	$0.000000208
512	$0.000000833
1024	$0.000001667

E.g. 30,000,000 Invocations at 128MB RAM for 200ms

=

$11.63

Costs of Serverless

- Developer time
- Monitoring services
- API Gateway HTTP calls
- Lambda invocations
- Database

Savings of Serverless

- DevOps
- SiteOps
- Capital expenses (CAPEX)
- Unused server or database capacity
- Monitoring and maintenance

Future Of DevOps

The End

Email me: **mischa@jetbridge.com**

mish.dev
spiegelmock.com
jetbridge.com