Microservices

A Tutorial Session
JavaOne 2016

by Jason Swartz

@swartzrock

CLASSPASS

MICROSERVICES

WHAT

WHY

a.k.a “WHEN”

HOW

What's A Microservice?

Your Application, Subdivided

Recipe: Microservices

(makes 5-10 microservices)

Ingredients:

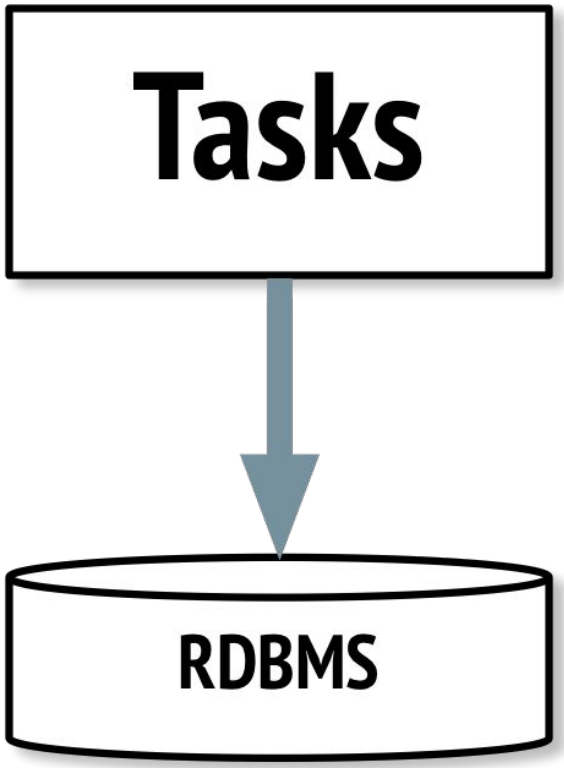
- A Monolithic API, Too Big To Handle

Instructions:

1. Find the seams in your application - by operation, business area, or team.
2. Split your API into multiple smaller API's, each on its own server.
3. Figure out how to deploy them w/o breaking everything.
4. Figure out how they'll talk to each other w/o breaking more things.
5. Figure out how complex data operations will work across multiple data stores.

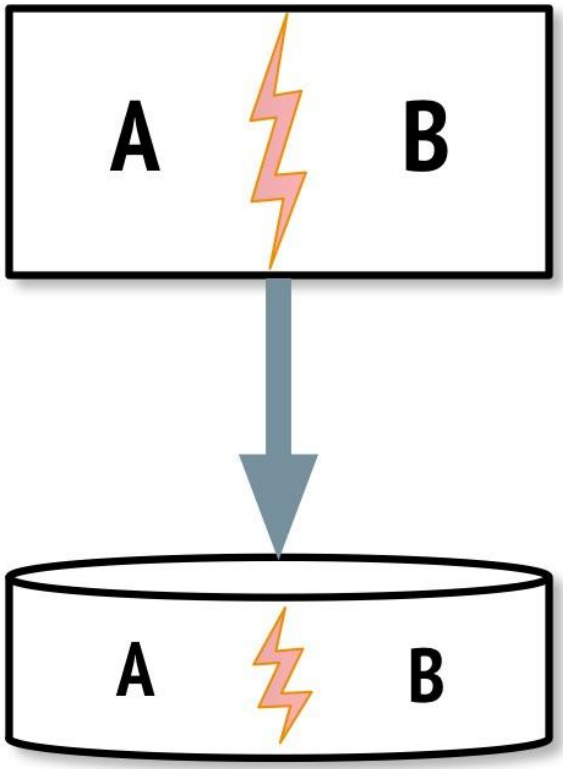
Tasks

Done	Title	Assignee
Yes	Install web server	Eric
No	Add tls certs	Ronnie
No	Config ELB	Pete


Tasks
RDBMS


Tasks
RDBMS
WARNING:
MONOLITH

Let's Split-Up That Service

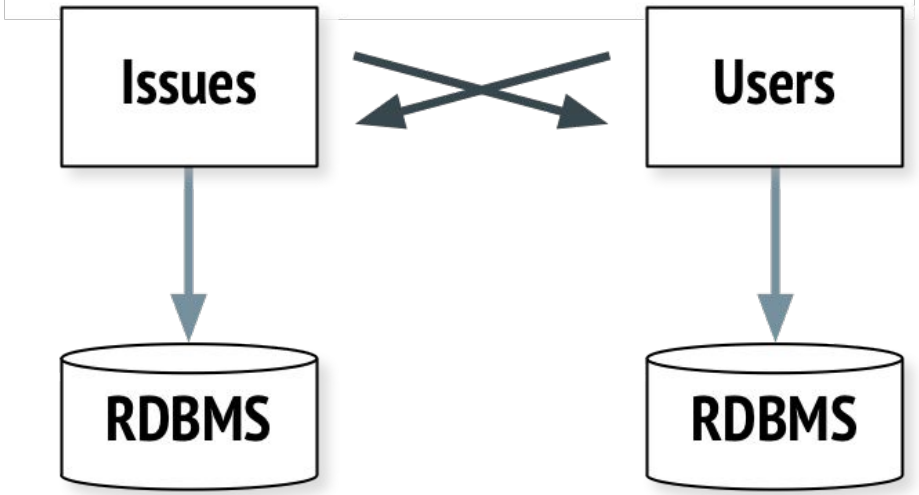

A
B
A
B

Tasks

Done	Title	Assignee
Yes	Install web server	Eric
No	Add tls certs	Ronnie
No	Config ELB	Pete

Tasks

Microservices!



Microservice Properties

1. Small & Maintainable
2. Private data
3. Generally Isolated

Microservice Nice-To-Haves

1. Small & Maintainable
2. Private data
3. Generally Isolated

Microservice Nice-To-Haves

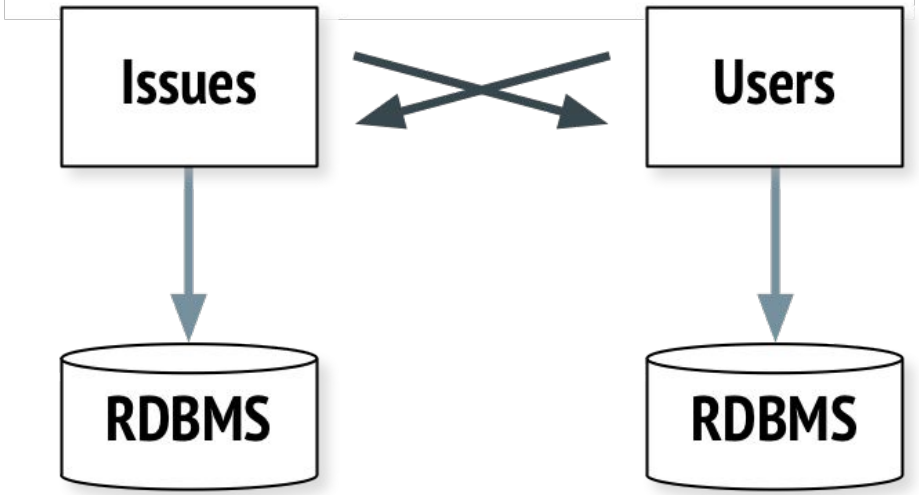
1. Findable
2. Scalable
3. Resilient
4. Monitorable

Microservice Real Nice-To-Haves

Microservices That Speak To Each Other Via Asynchronous Communication

Tasks

Microservices!



Let's See a Real Transaction

```
Web Client:      PUT issues/123 Assignee=10
Issues Service: Read from issues db
Issues Service: GET users/10
Issues Service: POST users/notify/10
Issues Service: Write to issues db
Issues Service: Return response with result
Web Client:      Read response & status then handle it
```

```
Web Client:      PUT issues/123 Assignee=10
Issues Service: Read from issues db
// block for synchronous transaction
// block for synchronous transaction
Issues Service: Write to issues db
Issues Service: Return response with result
Web Client:      Read response & status then handle it
```

Blocking Isn't Isolated, Scalable Or Resilient

But It Is Easy!

Synchronous Messaging Pros

1. Easy!
2. Gratifying.
3. Testable.

Synchronous Messaging Cons

1. Prevents microservices from being isolated, scalable, and resilient.
2. Hope you like waiting.

Microservices Should Talk Asynchronously

Talking Asynchronously

1. Non-Blocking IO (eg RxJava, Netty)
2. Queues (eg SQS)
3. Event Streams (eg Kafka, Kinesis)
4. Actors (eg Akka)

What About The Clients?

Responding Asynchronously

1. Synchronous REST with Async Clients
2. Http 202
3. Websockets
4. Http/2

That's What Microservices Are All About

WHAT

WHY

a.k.a “WHEN”

HOW

Monolith

To

Microservices

Why

Might This Be A Good Idea?

(Assuming You're Planning to Split Up your Monolith)

Monoliths - Not So Bad

1. Finding code and refactoring it is pretty easy.
2. No network lag
3. No format conversion errors
4. Deploy in one go

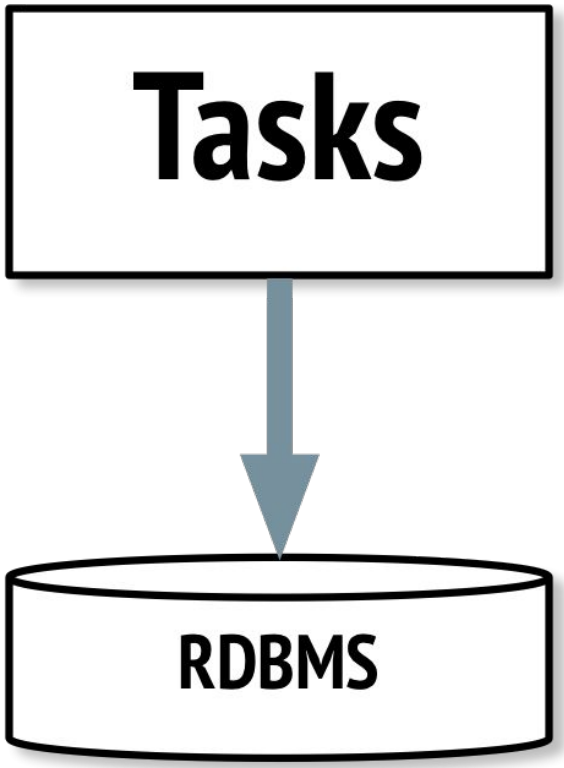
That's All

Thanks For Listening!

Monoliths - Not So Scalable Either

1. Hard to scale up just one feature
2. Hard to deploy just one change
3. Not suitable for more than a small team

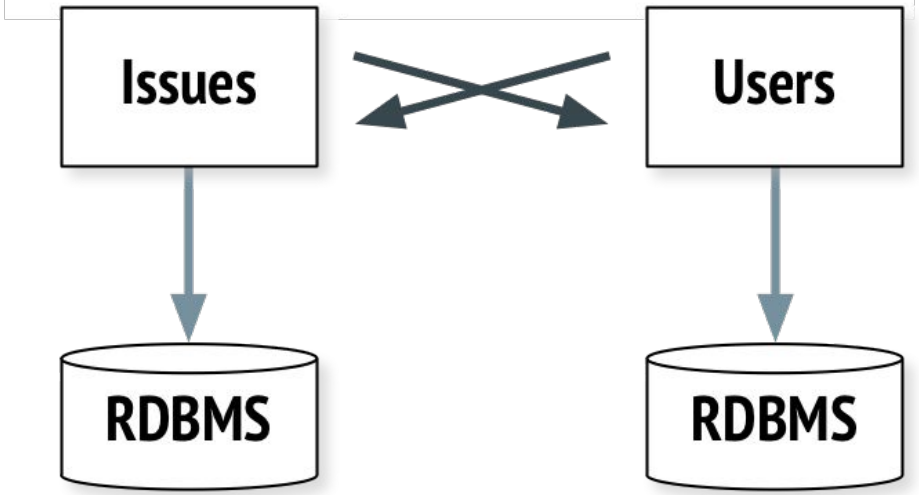
Organizations Are Constrained To Produce Designs That Match Their Communication Structure


Tasks
RDBMS

Max Capacity:
One Team

Tasks

Microservices!



Max Capacity:
Two Teams

Don’t Split Monoliths Across Your Communication Structures

Microservices Pros

(we kind of covered this already)

Microservices Challenges

1. Network Lag
2. Multiple Data Formats
3. Service Resolution
4. Troubleshooting
5. More Deployment Work
6. More Monitoring

No Problem.
Let's Get Started On
How
You'll Build Em

WHAT

WHY

a.k.a “WHEN”

HOW

Microservices In Java

How Will You Get Your App In Production?

Deployment Formats

1. Self-Running Jar’s
2. VM Images
3. Docker Images

Deployment Strategies

1. Use A CI To Build / Test / Merge in CI
2. Use A CD To Deploy (Push-Button / Auto)
3. VM Deployment (eg Spinnaker)
4. Docker Deployment (eg ECS, K8s, Marathon)
5. Cluster Management (eg K8s, Mesos, Swarm)

Should I Deploy To The Cloud?

Yes

Should I Test It First?

YES OH YES

Microservice Management

1. Service Resolution (K8s, Consul)
2. Scaling
3. Visibility
4. Authentication
5. Monitoring

Let's Talk Microservices Development

Choosing A Framework

(why switch now?)

Choosing A Framework

1. Spring Boot
2. Lagom
3. Ratpack
4. Dropwizard

Choosing A Data Store

1. Good Ol’ SQL (eg MySQL, Postgresql)
2. Document-type NoSQL
3. Fast Key-Value Caches

Choosing A Data Scheme

1. CRUD operations endpoints & tables
2. CRUD endpoints and immutable tables
3. Event-Sourced Data

Create-Read-Update-Delete

Let's Talk About
Immutable Tables

Tasks

Done	Title	Assignee
Yes	Install web server	Eric
No	Add tls certs	Ronnie
No	Config ELB	Pete

Issue Endpoints

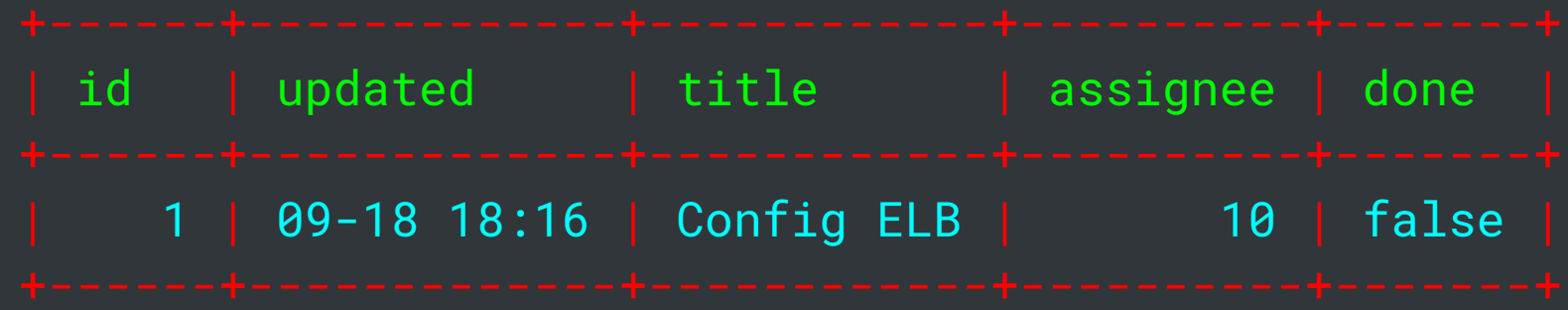
```
GET   /issues
GET   /issues/{id}
POST  /issues
PUT   /issues/{id}
```

How Do These Events Affect The Database?

1. POST /issues title='Config ELB'

id	updated	title	assignee	done
1	09-18 18:13	Config ELB	NULL	false

1. POST /issues title='Config ELB'
2. PUT /issues/1 assignee=10



id	updated	title	assignee	done
1	09-18 18:16	Config ELB	10	false

1. POST /issues title='Config ELB'
2. PUT /issues/1 assignee=10
3. PUT /issues/1 done=true

id	updated	title	assignee	done
1	09-18 18:24	Config ELB	NULL	true

1. POST /issues title='Config ELB'
2. PUT /issues/1 assignee=10
3. PUT /issues/1 done=true

Not Bad.

id	updated	title	assignee	done
1	09-18 18:24	Config ELB	NULL	true

1. POST /issues title='Config ELB'
2. PUT /issues/1 assignee=10
3. PUT /issues/1 done=true

Do You Know How We Got Here?

id	updated	title	assignee	done
1	09-18 18:24	Config ELB	NULL	true

Do You Know How We Got Here?

id	updated	title	assignee	done
1	09-18 18:24	Config ELB	NULL	true

1. POST /issues title='Config ELB'
2. PUT /issues/1 assignee=10
3. PUT /issues/1 done=true

Do You Know How We Got Here?

id	updated	title	assignee	done
1	09-18 18:24	Config ELB	NULL	true

1. POST /issues title='Config ELB'
2. PUT /issues/1 assignee=10
3. PUT /issues/1 done=true

Why is 'assignee' NULL?

id	updated	title	assignee	done
1	09-18 18:24	Config ELB	NULL	true

Mutable Table Rows Lose History

Immutable Table Rows Keep Their History

Let's Try To

Lock Down

Our State

1. POST /issues title='Config ELB'

id	updated	title	assignee	done
1	09-18 18:13	Config ELB	NULL	false

1. POST /issues title='Config ELB'
2. PUT /issues/1 assignee=10

id	updated	title	assignee	done
1	09-18 18:13	Config ELB	NULL	false
1	09-18 18:16	Config ELB	10	false

1. POST /issues title='Config ELB'
2. PUT /issues/1 assignee=10
3. PUT /issues/1 done=true

id	updated	title	assignee	done
1	09-18 18:13	Config ELB	NULL	false
1	09-18 18:16	Config ELB	10	false
1	09-18 18:19	Config ELB	NULL	false
1	09-18 18:24	Config ELB	NULL	true

1. POST /issues title='Config ELB'
2. PUT /issues/1 assignee=10
3. PUT /issues/1 done=true

id	updated	title	assignee	done
1	09-18 18:13	Config ELB	NULL	false
1	09-18 18:16	Config ELB	10	false
1	09-18 18:19	Config ELB	NULL	false
1	09-18 18:24	Config ELB	NULL	true

1. GET /issues/1

id	updated	title	assignee	done
1	09-18 18:13	Config ELB	NULL	false
1	09-18 18:16	Config ELB	10	false
1	09-18 18:19	Config ELB	NULL	false

id	updated	title	assignee	done
1	09-18 18:24	Config ELB	NULL	true

How Do You Make An Append-Only Table?

One: Don't Let Your DB User Make Changes

```
Grant select, insert on
issues to my-db-user;
```

Two: Pick The Right Columns

```
create table issues (
  id serial,
  created timestamp default now(),
  creator_id int,
  issue_id int default nextval('iseq'),
  title text,
  assignee int,
  done boolean default false
)
```

Do We Still Need State?

Let's Talk About
Event-Sourcing

1. POST /issues title='Config ELB'
2. PUT /issues/1 assignee=10
3. PUT /issues/1 done=true

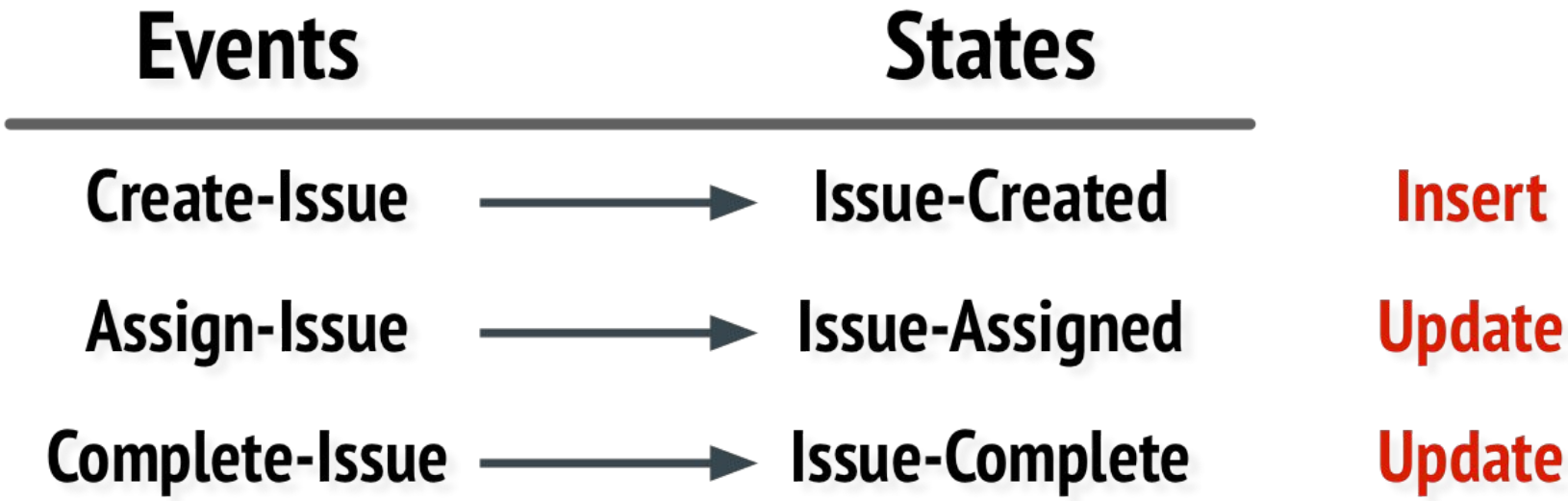
id	updated	title	assignee	done
1	09-18 18:13	Config ELB	NULL	false
1	09-18 18:16	Config ELB	10	false
1	09-18 18:24	Config ELB	10	true

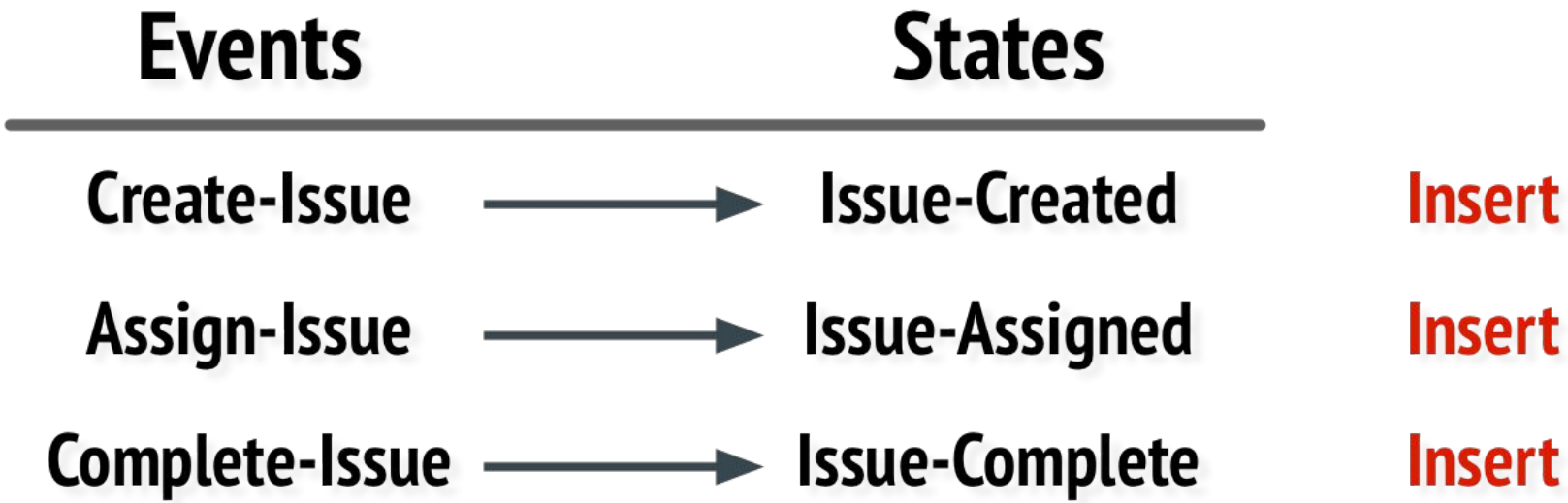
1. POST /issues title='Config ELB'
2. PUT /issues/1 assignee=10
3. PUT /issues/1 done=true

Events

States

id	updated	title	assignee	done
1	09-18 18:13	Config ELB	NULL	false
1	09-18 18:16	Config ELB	10	false
1	09-18 18:24	Config ELB	10	true


Events
States
Create-Issue
Issue-Created
Insert
Assign-Issue
Issue-Assigned
Update
Complete-Issue
Issue-Complete
Update


Events
States
Create-Issue
Issue-Created
Insert
Assign-Issue
Issue-Assigned
Insert
Complete-Issue
Issue-Complete
Insert


Events
States
Create-Issue
Insert
Issue-Created
Assign-Issue
Insert
Issue-Assigned
Complete-Issue
Insert
Issue-Complete


Events
Virtual States
Create-Issue
Insert
Issue-Created
Assign-Issue
Insert
Issue-Assigned
Complete-Issue
Insert
Issue-Complete

Now We're Storing Events, Not States

```
create table issue_events (
  id serial,
  created timestamp default now(),
  issue_id int default nextval('iseq'),
  originator text,
  payload text
)
```

1. POST /issue/1/event ‘Originator: 4a48239-8a..’
   payload=’<Update val=”done=true”>’

id	created	issue_id	originator	payload
14	09-18 18:50	1	4a482...	<...>

Create Events And Simulate The State

Real Events

1. Create-Issue

Virtual States

```
Issue(“Config ELB”, null, false);
```

Real Events

1. ~~Create-Issue~~
2. Assign-Issue

Virtual States

```
Issue(“Config ELB”, 10, false);
```

Real Events

1. ~~Create-Issue~~
2. ~~Assign-Issue~~
3. Complete-Issue

Virtual States

```
Issue(“Config ELB”, 10, true);
```

So Why Use Event-Sourcing?

Reasons For Event-Sourcing

1. High Write Performance
2. Potential for Command/Query Separation
3. Auditable
4. Replayable
5. Undo-able
6. Monitorable

It's Like Having Control Over The Versions Of Your State Changes

It's Like Having Control Over The Versions Of Your Data

It's Like Git For Your Data

Reasons Against Event-Sourcing

1. Frankly, It’s Weird
2. Requires Events. No Events, No Event-Sourcing.
3. As Of Sept 2016, It’s Still Non-Standard

That About Sums Up

Microservice Development

That About Sums Up

Microservice Development

Okay, Actually That's The Entire Session

Unless There's More Time

WHAT

WHY

a.k.a “WHEN”

HOW

by Jason Swartz

@swartzrock

CLASSPASS

Microservices

A Tutorial Session
JavaOne 2016

Make Sure To See...

Web Applications for the REST of Us: An Introduction to Ember.js, Akka, and Spray (Craig Tataryn & Sean Kowaski)

A Practical RxJava Example with Ratpack (Laurent Doguin)

One Microservice Is No Microservice: They Come In Systems (Markus Eisele)

Thank You For Attending

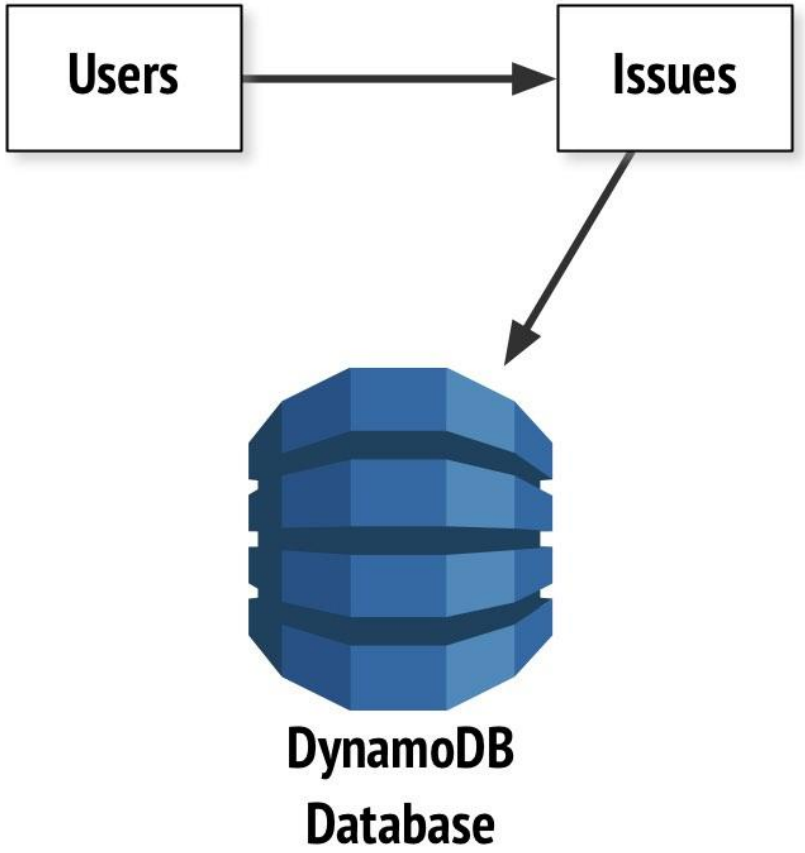
Fin

Just Kidding, There's

Another Section

Let's Talk Asynchronous Architectures in AWS

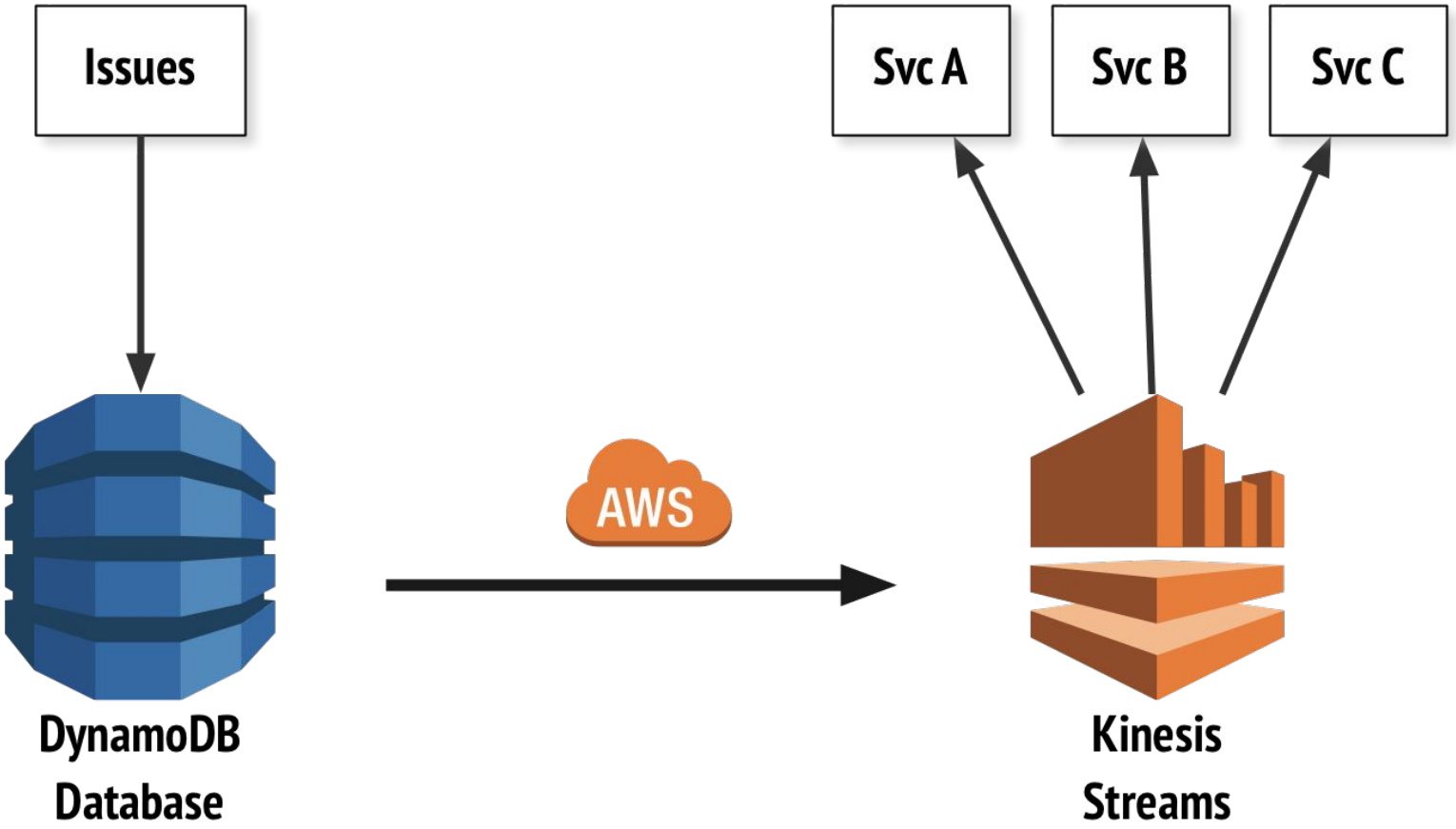
Do You Want To Use Caches To Prevent Unnecessary Blocking Calls?


Users
Issues
DynamoDB
Database

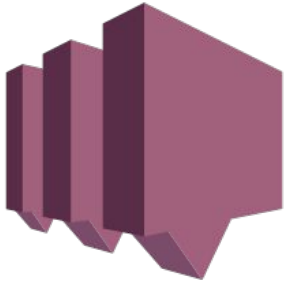
Do You Want To Insert Rows And Publish Events In One Stroke?

DynamoDB
Database

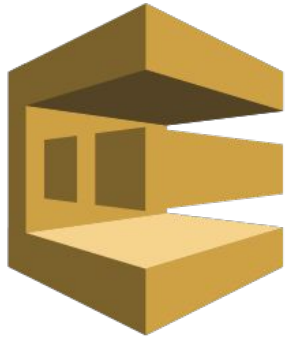
Kinesis
Streams


Issues
Svc A
Svc B
Svc C
AWS
DynamoDB
Database
Kinesis
Streams

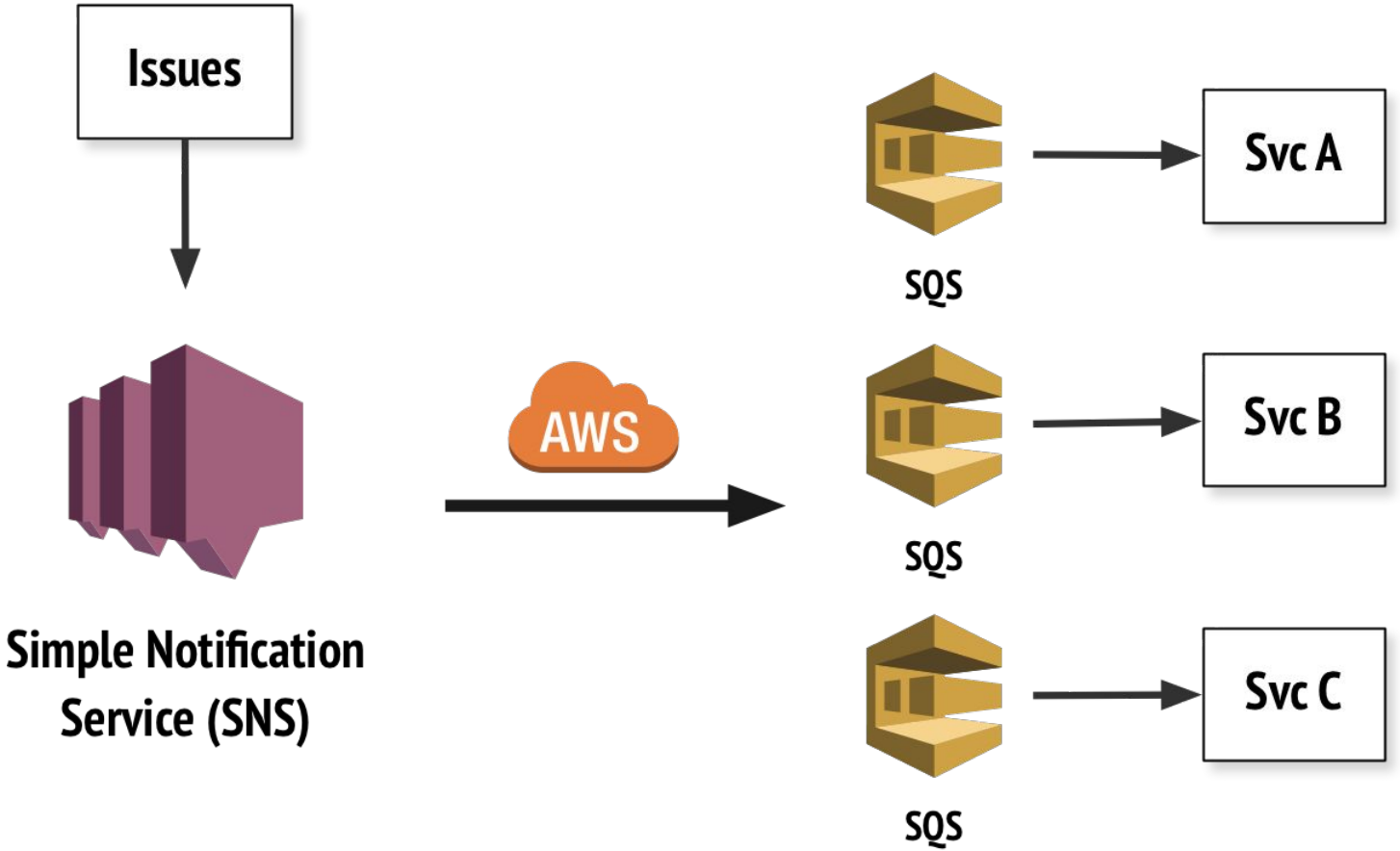
Do You Want To Mass-Publish One Event To All Microservices?



Simple Notification Service (SNS)



Simple Queue Service (SQS)


Issues
Simple Notification Service (SNS)
AWS
SQS
SQS
SQS
Svc A
Svc B
Svc C

Note About Your Solution: I'm Not Saying It's AWS

But It's AWS

Okay, We Are Really Done This Time

THIS SPACE
LEFT BLANK