Learning the Three Types of Microservices

SACon London 2018

Mike Amundsen
@mamund
API Academy
Mike Amundsen
@mamund
COMPLIMENTARY O'REILLY BOOK: SECURING MICROSERVICE APIs

40+ PAGES OF PRACTICAL GUIDANCE FOR SUSTAINABLE AND SCALABLE ACCESS CONTROL
Microservice Architecture: Aligning Principles, Practices, and Culture

Microservices is the next evolution in software architecture designed to help organizations embrace continual change in the digital economy. But how do you design and apply an effective microservice architecture?

This new book from O’Reilly provides comprehensive guidance through seven valuable chapters that give you a deep-dive into:

- The benefits and principles of microservices
- A design-based approach to microservice architecture
- Lessons for applying microservices in practice

http://g.mamund.com/msabook
@TheCAMBook

"A reusable guide to the technology, business, and politics of doing APIs at scale within the enterprise."

-- Kin Lane, API Evangelist
Continuous API Management: Making the Right Decisions in an Evolving Landscape
1st Edition
by Mehdi Medjaoui (Author), Erik Wilde (Author), Ronnie Mitra (Author), Mike Amundsen (Author)

Paperback $49.99
Pre-order
This title will be released on January 4, 2019.
Ships from and sold by Amazon.com. Gift-wrap available.

Other Sellers from $49.99

More Buying Choices
1 New from $49.99

ISBN-10: 1492043559
Why is ISBN important? →
Overview

● Programming the Network
● Microservices
● Three Types of Microservice Components
● Nygard's Stability Patterns
● Applying Nygard to Microservices
● But Wait, There's More...
Overhead wires are down close to Paddington. Please refer to @TfLRail @GWRHelp @HeathrowExpress @nationalrailenq for latest information.

Emergency service are on route inc @networkrail. Please do not leave trapped trains, remain on board. Rail Tracks are very dangerous.

12:34 PM - Oct 28, 2018

17 people like this 38 people are talking about this
@GWRHelp Stuck at Hayes & Harlington. Will be missing the NFL game at Wembley that we’ve been looking forward to for months. Will we be compensated for those tickets as well as our train tickets? No one here at Hayes & Harlington to help stranded passengers either. Disgraceful.
12:24 PM - Oct 28, 2018

1 like. See Kate Griffiths's other Tweets
Kate Griffiths
@kgriffiths__

@GWRHelp Stuck at Hayes & Harlington. Will be missing the NFL game at Wembley that we’ve been looking forward to for months. Will have to catch a train in addition to our train ticket to Harlington to help.

12:24 PM - Oct 28, 2018

XandraE
@XandraE1

Replying to @HeathrowExpress

My husband and I are going to miss our flight back to Atlanta which means I need a new flight and accommodations. Have the airlines been notified? Is there a customer service team I can call

12:39 PM - Oct 28, 2018
Kate Griffiths  
@kgriffiths__  
@GWRHelp Stuck at Hayes & Harlington for months. Will our trains be running to Wembley as well as our train tickets being refunded?  
12:24 PM - Oct 28, 2018  

Joshua Newell  
@XandraE  
Replying to @He671  
@kgriffiths__  
My husband and I are based in Atlanta which makes it hard to find accommodation in the UK. Is there a customer service team I can call?  
12:39 PM - Oct 28, 2018  

Jen Kenward  
@JenKenward  
#Hellomynameis Jen  
Total @networkrail disaster trains to Paddington out again, offloaded at Reading & now herded like cattle into an all stops train to Waterloo, barely standing room (not breathing) only - total nightmare, no support or respect for passengers  
3:24 PM - Oct 28, 2018 · Reading, England  

20 18 people are talking about this
@GWRHelp Stuck at Hayes & Harlington - the NFL game at Wembley that we've all been looking forward to for months. Will we ever see a return to normal?— @kgriffiths__

#HellomynamesJen @JenKenward

Total @networkrail disaster trains to Paddington out again, offloaded at Reading & now herded like cattle into an all stops train to Waterloo, barely standing room (not even a 1/2) - lost 15 mins and less respect for the staff. 😞

Ross Whittaker @RossWhittaker5

Not kidding, looks like people are getting off onto the tracks before Paddington and walking through a work site out 😞 thankfully stocked up on supplies @GWRHelp

11:59 AM - Oct 28, 2018

See Ross Whittaker's other Tweets
Traveling
Traveling
Traveling the Network
Programming the Network
RedBoot(tm) bootstrap and debug environment [RAM]
Panasonic Avionics Corporation) release, version ("560328-212" v "1.07" b "0126"
) - built 15:35:59, May 22 2013

Platform: SM-02 (i386)
Copyright (C) 2000, 2001, 2002, Red Hat, Inc.

RAM: 0x00000000-0x000a0000, 0x00100000-0x01000000 available
Current Boot Count is 0

verifying MBR... Fix MBR:
Partition 0: already exists
Partition 1: already exists
Partition 2: already exists
Partition 3: already exists

verifying image... OK.
== Executing kernel in 5 seconds - enter 'c' to abort
Load Address 0x00000000
Image length 0x00e2f5d5
Loading kernel binary...
Read image signature... if 8b
Decompressing image...
Programming the Network

"There is no simultaneity at a distance."

-- Pat Helland (2005)
Newton rules the "inside"
Einstein rules the "outside"
Programming the Network

There is no simultaneity at a distance!
-- Similar to the speed of light bounding information
-- By the time you see a distant object, it may have changed!
-- By the time you see a message, the data may have changed!

Pat Helland

Programming the Network

There is no simultaneity at a distance!
-- Similar to the speed of light bounding information
-- By the time you see a distant object, it may have changed!
-- By the time you see a message, the data may have changed!

Services, transactions, and locks bound simultaneity!
-- Inside a transaction, things are simultaneous
-- Simultaneity exists only inside a transaction!
-- Simultaneity exists only inside a service!

Programming the Network

Fallacies of Distributed Computing (1994)

1. The network is reliable.
2. Latency is zero.
3. Bandwidth is infinite.
4. The network is secure.
5. Topology doesn't change.
6. There is one administrator.
7. Transport cost is zero.
8. The network is homogeneous.

The Language of the System (2012)

Rich Hickey

The Stacks

<table>
<thead>
<tr>
<th>Program</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application libs</td>
<td>Application as services</td>
</tr>
<tr>
<td>Runtime and core libs</td>
<td>Simple Services</td>
</tr>
<tr>
<td>Language primitives</td>
<td>Protocols and formats</td>
</tr>
</tbody>
</table>

Programming the Network brings new challenges
Microservices
"An approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms."

-- Martin Fowler, 2014

https://www.thoughtworks.com/insights/blog/microservices-nutshell
"Emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components."

-- Roy Fielding, 2000
"A universal linked information system, in which generality and portability are [most] important."

-- Tim Berners-Lee, 1989
Microservice Characteristics

- Make each program to one thing well
- Expect the output of every program to be the input of another program
- Design and build software to be tried early
- Use tools to lighten the programming task
Unix Operating Principles (1978)

- Make each program to one thing well
- Expect the output of every program to be the input of another program
- Design and build software to be tried early
- Use tools to lighten the programming task

Loosely-coupled components running in an engineered system.
Three Types of Microservices
Three Types of Microservices

- Stateless
- Persistence
- Aggregator
Stateless Microservices
Stateless Microservices

- Simple processors (converters, translators, etc.)
- No dependence on other microservices
- No local data storage (disk I/O)

The most common MSC example, but the least useful!
Stateless Microservices

- No shared state
- Easy to replace
- Easy to scale up

Ephemeral Computing
Stateless Microservices

```javascript
// http server handling data conversions
function conversionServer(request, response) {
    response = convertValue(request);
    return response;
}
```

WARNING: NOT REAL CODE!
Persistence Microservices
Persistence Microservices

- Simple (local) storage (reads and/or writes)
- Disk I/O dependent
- Possibly VM or one-U dependent

Commonly needed MSC, not the easiest to implement.
Persistence Microservices

- System of Record/Source of Truth
- Relatively easy to scale for reads (CQRS)
- No cross-service two-phase commits (Saga)

Durable Storage
Persistence Microservices

```javascript
function updateOrders(request, response) {
    response = localStorage.write(request);
    return response;
}
```

WARNING: NOT REAL CODE!
Aggregator Microservices
Aggregator Microservices

- Depends on other ("distant") microservices
- Network dependent
- Usually Disk I/O dependence, too

The most often-needed; most challenging, too.
Aggregator Microservices

- Sequence vs. Parallel calls
- Timing is everything
- Easy to scale (should be…)

Workflow Choreography
Aggregator Microservices

```javascript
function writeOrders(request, response) {
  var resourceList = ["customerDB", "orderDB", "salesDB"]
  var serviceList = gatherResources(resourceList);
  response = serviceList(request);
  return response;
}
```

WARNING: NOT REAL CODE!
Three Types of Microservices

- Stateless (ephemeral)
- Persistence (durable)
- Aggregator (workflow)
But, what about the network?
Nygard's Stability Patterns
“Bugs will happen. They cannot be eliminated, so they must be survived instead.”

-- Michael T. Nygard
Release It!
Second Edition
Design and Deploy
Production-Ready Software
Nygard Stability Patterns

- Timeout
- Circuit Breaker
- Bulkhead
- Steady State
- Fail Fast
- Handshaking
"Nygard Stability Patterns" -- Timeout

"The timeout is a simple mechanism allowing you to stop waiting for an answer once you think it will not come."

-- Chapter 5.1
"The timeout is a simple mechanism allowing you to stop waiting for an answer once you think it will not come."

-- Ch 5.1

```
// set up proper shutdown
process.on('SIGTERM', function () {
    discovery.unregister(null, function(response) {
        try {
            uuidGenerator.close(function() {
                console.log('gracefully shutting down');
                process.exit(0);
            })
        } catch(e){}
    });
    setTimeout(function() {
        console.error('forcibly shutting down');
        process.exit(1);
    }, 10000);
});
```
"Circuit breakers are a way to automatically degrade functionality when the system is under stress."

-- Chapter 5.2
"Nygard Stability Patterns" -- Circuit Breaker

"Circuit breakers are a way to automatically degrade functionality when the system is under stress."

-- Chapter 5.2
"Nygard Stability Patterns" -- Bulkhead

"The bulkhead enforces a principle of damage containment."

-- Chapter 5.3
"Nygard Stability Patterns" -- Bulkhead

"The bulkhead enforces a principle of damage containment."
-- Chapter 5.3
"Nygard Stability Patterns" -- Bulkhead

"The bulkhead enforces a principle of damage containment."

-- Chapter 5.3
"Nygard Stability Patterns" -- Steady State

"The system should be able to run indefinitely without human intervention."

-- Chapter 5.4

- Avoid fiddling
- Purge data w/ app logic
- Limit caching
- Roll the logs
"Nygard Stability Patterns" -- Steady State

"The system should be able to run indefinitely without human intervention."
-- Chapter 5.4

- Avoid fiddling
- Purge data w/ app logic
- Limit caching
- Roll the logs
"Nygard Stability Patterns" -- Fail Fast

"If the system can determine in advance that it will fail at an operation, it’s always better to fail fast."

-- Chapter 5.5
"Nygard Stability Patterns" -- Fail Fast

"If the system can determine in advance that it will fail at an operation, it’s always better to fail fast."

-- Chapter 5.5

```javascript
function bookOrders(orderList, timeBudget) {
    var status = false;
    var resources = ["customerdata","orderdata","salesdata"];
    setTimeout(function(resources) {
        var status = confirmResourceAvailability(resources);
        if(status===true && timeBudget>500) {
            try {
                status = writeOrders(orderList,resources);
            }
            catch (ex) {
                error("failed to write orders : {errordcode}",ex);
            }
        }
        else {
            error("failed to acquire resources : FAILFAST");
        }, timeBudget);
    },
```
"Handshaking is all about letting the server protect itself by throttling its own workload."

-- Chapter 5.6
"Nygard Stability Patterns" -- Handshaking

"Handshaking is all about letting the server protect itself by throttling its own workload."

-- Chapter 5.6

```
function sendOrders(orderList, timeBudget) {
    if ((health.responseMS+health.latencyMS) < timeBudget ) {
        bookOrders.send(orderList,timeBudget);
    } else {
        error("failed to send orders: HEALTHCHECK");
    }
}
```

WARNING: NOT REAL CODE!
"Nygard Stability Patterns" -- Cache

"Caching can reduce the load on the server and cut response times to a fraction of what they would be without caching."

-- Chapter 10.2
"Nygard Stability Patterns" -- Cache

"Caching can reduce the load on the server and cut response times to a fraction of what they would be without caching."

-- Chapter 10.2

```javascript
// server marks data cache-able
function sendResults(response) {
    response.writeHead(status,
    { 'Content-Type': 'text/plain',
      'Cache-Control': 'public,max-age=108000'
    );
    response.end(value+'
');
}
```
"Nygard Stability Patterns" -- Cache

"Caching can reduce the load on the server and cut response times to a fraction of what they would be without caching."

-- Chapter 10.2

```
// server marks data cache-able
function sendResults(response) {
    response.writeHead(status,
    { 'Content-Type': 'text/plain',
    'Cache-Control': 'public,max-age=3600' });
    response.end(value+
    };
```

```
// client manages local cache
function getData(URL) {
    data = null;
    data = cache.read(URL);
    if(!data) {
        data = requestResults(URL);
        cache.write(URL,data);
    }
    return data;
```

WARNING: NOT REAL CODE!
Stabilizing Stateless Microservices
Stateless Microservices

```javascript
// http server handling data conversions
function conversionServer(request, response) {
    response = convertValue(request);
    return response;
}
```
Networked Stateless

- What if the work takes too long?
Stable Stateless Microservices

1. Fail-Fast

```javascript
// http server handling data conversions
function conversionServer(request, response) {
    if (request.timeBudget > my.averageResponse) {
        response = FailFastError(request);
    } else {
        response = convertValue(request);
    }
    return response;
}
```
Stabilizing Persistence Microservices
Persistence Microservices

```javascript
function updateOrders(request, response) {
  response = localStorage.write(request);
  return response;
}
```

WARNING: NOT REAL CODE!
Networked Persistence

- What if the work takes too long?
- What is the dependent service doesn't respond in time?
- What if the dependent service is down?
- What if the storage overflows (data, logs, etc.)?
Stable Persistence Microservices

```javascript
function updateOrders(request, response) {
    if (request.timeBudget < localStorage.latency) {
        response = FailFastError(request);
    } else {
        response = setTimeout(circuitBreaker(
            localStorage.write(request),
            {timeout: 10, maxFail: 3, reset: 30}
        ), timeBudget);
    }
    return response;
}
```

1. Fail-Fast
2. Timeout
3. Circuit Breaker
4. Steady State
Stabilizing Aggregator Microservices
Aggregator Microservices

```javascript
function writeOrders(request, response) {
    var resourceList = ['customerDB', 'orderDB', 'salesDB']
    var serviceList = gatherResources(resourceList);
    response = serviceList(request)

    return response;
}
```

WARNING: NOT REAL CODE!
Networked Aggregators

- What if the work takes too long?
- What if a dependent services doesn't respond in time?
- What if a dependent service is down?
- What if storage overflows (data, logs, etc.)?
- What if a dependent service is unhealthy?
- What if traffic for a service spikes?
Stable Aggregator Microservices

function writeOrders(request, response) {
    var resourceList = ["customerDB", "orderDB", "salesDB"]

    setTimeout(function(request, response, resourceList) {
        var serviceList = gatherResources(resourceList);
        if(serviceList.estimatedCost > request.timeBudget) {
            response = FailFast(request);
        } else {
            if(serviceList.healthy === true) {
                circuitBreaker(serviceList, request,
                {timeout:10,maxFail:3,reset:30});
            }
        }
},request.timeBudget);

return response;
}

1. Fail-Fast
2. Timeout
3. Circuit Breaker
4. Steady State
5. Handshaking
6. Bulkhead

WARNING: NOT REAL CODE!
Joe asks:

Is All This Clutter Really Necessary?

You may think, as I did when porting the sockets library, that handling all the possible timeouts creates undue complexity in your code. It certainly adds complexity. You may find that half your code is devoted to error handling instead of providing features. I argue, however, that the essence of aiming for production—instead of aiming for QA—is handling the slings and arrows of outrageous fortune. That error-handling code, if done well, adds resilience. Your users may not thank you for it, because nobody notices when a system doesn’t go down, but you will sleep better at night.
Applying Nygard's Patterns to Services

- **Stateless**
  - *fail fast*

- **Persistence**
  - *fail fast, timeout, circuit breaker, steady state*

- **Aggregation**
  - *fail fast, timeout, circuit breaker, steady state, handshaking, bulkhead*

Apply Nygard's Stability Patterns to improve the health of your components and your system.
BUT WAIT, there's more!
Aim for Interop, not Integration...

"Interoperation is peer to peer. Integration is where a system is subsumed within another."

-- Michael Platt, Microsoft

Aim for Interop, not Integration...
Include time/distance in your models

"There is no simultaneity at a distance."

-- Pat Helland, Salesforce
Include time/distance in your models

"I'm sorry that coined the term 'objects' for this topic. The big idea is 'messaging'."

Alan Kay, 1998
Remember, you're programming the network

1. The network is reliable.
2. Latency is zero.
3. Bandwidth is infinite.
4. The network is secure.
5. Topology doesn't change.
6. There is one administrator.
7. Transport cost is zero.
8. The network is homogeneous.

Remember, you're programming the *network*

- **Safety**
Safety

The HTTP protocol supports a number of "safe" actions such as HEAD, and GET.

The HTTP methods PUT, POST, and DELETE are categorized as "unsafe" actions.

Remember, you're programming the network.
Remember, you're programming the network

- Safety
- Idempotence

Remember, you're programming the *network*

- Safety
- Idempotence

In HTML when a FORM element has the METHOD property set to "get" this represents an idempotent action.

When the same property is set to "post" the affordance represents a non-idempotent action.

Other Considerations...

- Interop vs. Integration
- Time & Distance
- Safety & Idempotence
So...
We need microservices...
So that we can program the network...
Which means applying patterns to our code...

```javascript
function writeOrders(request, response) {
    var resourceList = ['customerDB', 'orderDB', 'salesDB']

    setTimeout(function(request, response, resourceList) {
        var serviceList = gatherResources(resourceList);
        if(serviceList.estimatedCost > request.timeBudget) {
            response = FailFast(request);
        } else {
            if(serviceList.healthy === true) {
                circuitBreaker(serviceList, request, {
                    timeout: 10, maxFail: 3, reset: 30
                });
            }
        }, request.timeBudget);

    return response;
}
```

1. Fail-Fast
2. Timeout
3. Circuit Breaker
4. Steady State
5. Handshaking
6. Bulkhead
And that means understanding the role of semantics...
And the role of distance & time...

No Notion of “Now” in Between Services!
And constantly reminding ourselves of the challenge.
That's a lot!
The Best Software Architecture

"The best software architecture 'knows' what changes often and makes that easy."

- Paul Clements
Let's continue the conversation...
Learning the Three Types of Microservices

SACon London 2018

Mike Amundsen
@mamund
API Academy
Learning the Three Types of Microservices

SACon London 2018

Mike Amundsen
@mamund
API Academy