Microservices on AWS

AWS Summit Berlin 2016

Matthias Jung, Solutions Architect
Julien Simon, Evangelist

April, 12th, 2016
Agenda

What are Microservices?
Why Microservices?
Challenges of Microservices
Microservices on AWS
Docker with ECR & ECS - Demo
What are Microservices?
What are Microservices?

Google Trends
What are Microservices?

Related concepts
• Service Oriented Architectures
• API First
• Agile Software Development
• Continuous Delivery
• DevOps
Characteristics of Microservice Architectures

- Decentralized
- Independent
- Polyglot
- Do one thing well
- Black Box

You build it, you run it
Why Microservices?
Why Microservices?

Gilt: “From Monolith Ruby App to Distributed Scala Micro-Services” (NYC Tech Talks) [Link]

Nike: “Nike’s Journey to Microservices” (AWS Re:Invent 2014) [Link]

SoundCloud: ”Building Products at SoundCloud - Part III: Microservices in Scala and Finagle” [Link]

Capital One: “Lack Of Legacy Lets Capital One Build Nimble Infrastructure” [Link]

Hailo: “A Journey into Microservices” [Link]

Autoscout24: “Why Autoscout24 changes its technology” [Link]

Zalando: “From Monolith to Microservices” [Link]
Problems of Monolithic Architectures

Code complexity and maintainability
Deployment becomes the bottleneck
Fear to change
Lack of ownership
Failure dependencies
One size doesn’t fit all (ex: relational DB)
Hard to scale out
Problems of Monolithic Architectures

Code complexity and maintainability

Deployment becomes the bottleneck

Fear to change

Lack of ownership

Failure dependencies

One size doesn’t fit all (ex: relational DB)

Hard to scale out
Problems of Monolithic Architectures
Development Life Cycle with Small Teams
Benefits of Microservices

Speed
  • Faster development and deployment

Innovation
  • Autonomy of teams, culture of change
  • Ownership and DevOps culture

Quality
  • Composability and reusability
  • More maintainable code
  • Better scaling and optimizations
  • Failure Isolation and Resiliency
What Customers Say

“Avoid fear to change things”

“Applied SE best practices to operations”

“Easily switch between synchronous and asynchronous communication”

“Easy to start new things from scratch”

“People take ownership”

“Deploy more – deploy faster – deploy better code”
The Amazon DevOps Story
Service-Oriented Architecture (SOA)

Everything gets a service interface

Primitives

“Microservices”
Decentralized

Two-pizza teams

Agility, autonomy, accountability, and ownership

“DevOps”
Decentralized Ownership
Promote Best Practices
No gatekeepers
Support Agile
SW Dev Lifecycle
Technology Agnostic
# Deployments at Amazon.com

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean time between deployments (weekday)</td>
<td>~11.6s</td>
</tr>
<tr>
<td>Max number of deployments in a single hour</td>
<td>~1,079</td>
</tr>
<tr>
<td>Mean number of hosts simultaneously receiving a deployment</td>
<td>~10,000</td>
</tr>
<tr>
<td>Max number of hosts simultaneously receiving a deployment</td>
<td>~30,000</td>
</tr>
</tbody>
</table>
Challenges of Microservices
Challenges of Microservices

Complexity in Code Base

Complexity in Interactions
Complexity in Interactions

- Authentication?
- Eventual Consistency?
- Message Routing?
- Discovery?
- Monitoring?
- Cascading Failures?
- Delay?
- Transactions?
- Testing?
- API Management?
Challenges of Microservices

One size doesn’t fit all

Heterogeneity
No Standards
Challenges

Organizational Cultural Challenges

- You built it, you run it

Architectural Challenges

- Dealing with asynchronicity
- Cascading failures
- Discovery and authentication of services
- Integration Tests

Operational Challenges

- Duplication of processes and tools
- Complexity moves from components to interactions
- Debugging across components
- Deployment
Microservices on AWS
How Can AWS Help with **Operational Complexity**?

- **On Demand Resources**
  - no capacity guessing
  - resources in any size
  - parallel environments
How Can AWS Help with **Operational Complexity**?

- On Demand Resources
- Managed Services
Storage Options in the Traditional World

- Struct. Data
- Search Indices
- Meta Data
- Event Logs
- FS Blocks
- Temp Files
- Data Blobs
- Static Assets
- Back-Ups

Relational Database

NFS
Storage Options in the Cloud

- Struct. Data
- Search Indices
- Meta Data
- Event Logs
- FS Blocks
- Temp Files
- Data Blobs
- Static Assets
- Back-Ups

Amazon RDS
Amazon CloudSearch
DynamoDB
Amazon Kinesis
Amazon EBS
Ephemeral EC2 Storage
Amazon S3
CloudFront
Amazon Glacier
Don’t Reinvent the Wheel

If you find yourself writing your own…

Notification system
E-Mail component
Search engine
Workflow engine
Queue
Transcoding system
Monitoring system

…take a deep breath and stop it now!
How Can AWS Help with **Operational Complexity**?

- On Demand Resources
- Managed Services
- **Built-in features**
  - Monitoring via CloudWatch
  - Security: IAM, CloudTrail, KMS, …
  - Logging: CloudWatch Logs
  - Scalability: Auto-Scaling, ELB, S3, …
  - Availability: multiple Availability Zones
Amazon ECS

AWS Lambda

Amazon EC2

Amazon CloudWatch Logs

Amazon Kinesis Firehose

Amazon Redshift

Amazon QuickSight
How Can AWS Help with Operational Complexity?

• On Demand Resources
• Managed Services
• Built-in features
  • monitoring, security, logging, …
  • scalability, availability, …
• Everything Programmable

AWS CLI & SDKs
How Can AWS Help with **Operational Complexity**?

- On Demand Resources
- Managed Services
- Built-in features
  - monitoring, security, logging, …
  - scalability, availability, …
- Everything Programmable
- Infrastructure as Code
How Can AWS Help with **Operational Complexity**?

- On Demand Resources
- Managed Services
- Built-in features
  - monitoring, security, logging, …
  - scalability, availability, …
- Everything Programmable
- Infrastructure as Code
- No Servers

AWS Lambda
How Can AWS Help with Operational Complexity?

- Run code without infrastructure
- Backend at any scale
- No administration
- JavaScript, Java, and Python

AWS Lambda
How Can AWS Help with Managing APIs?

- Managing multiple versions and stages?
- Monitoring 3rd party developer access?
- Access authorization?
- Traffic spikes?
- Caching?
How Can AWS Help with Managing APIs?

- Managing multiple versions and stages
- Monitoring 3rd party developer access
- Access authorization
- Traffic spikes
- Caching
- Swagger Support
- Request/Response Transformation
- API Mocking
How Can AWS Help with Scaling Deployments?

Code Commit

Code Pipeline

Elastic Beanstalk

OpsWorks

Code Deploy

Cloud Formation

Cloud Watch
Microservice Architectures
A Typical Microservice Architecture on AWS

- **Content Delivery**
  - CloudFront
  - S3
  - Static Content

- **API Layer**
  - Elastic Load Balancing
  - EC2
  - Auto Scaling group

- **Application Layer**
  - Elastic Load Balancing
  - EC2
  - Auto Scaling group

- **Persistency Layer**
  - ElastiCache
  - RDS
A Typical Microservice Architecture on AWS

Content Delivery
- CloudFront
- S3

API Layer
- API Gateway

Application Layer
- Elastic Load Balancing
- EC2
- Auto Scaling group

Persistency Layer
- ElastiCache
- RDS

Layers:
- Static Content (S3)
- API Gateway
- Elastic Load Balancing
- EC2
- Auto Scaling group
- ElastiCache
- RDS
A Typical Microservice Architecture on AWS

Content Delivery
- CloudFront
- Static Content

API Layer
- API Gateway
- S3

Application Layer
- Elastic Load Balancing
- EC2 Container Service
- EC2
- Auto Scaling group

Persistency Layer
- ElastiCache
- RDS
A Typical Microservice Architecture on AWS

- **Content Delivery Layer**: CloudFront
- **API Layer**: API Gateway
- **Application Layer**: Elastic Load Balancing, EC2 Container Service
- **Persistency Layer**: DynamoDB

- **Static Content**: S3
- **Auto Scaling Group**: EC2
A Typical Microservice Architecture on AWS

Content Delivery

API Layer

Application Layer

Persistency Layer

CloudFront

API Gateway

AWS Lambda

DynamoDB

Static Content

S3
A Typical Microservice Architecture on AWS

Content Delivery

API Layer

Application Layer

Persistency Layer

CloudFront

API Gateway

AWS Lambda

DynamoDB

Static Content

Serverless!
Docker with ECR & ECS - Demo