



AWS SERVERLESS AND MICROSERVICES





PACTERA IS A **GLOBAL** TECHNOLOGY
SERVICES COMPANY THAT DESIGNS,
BUILDS AND OPTIMIZES **DIGITAL**
APPLICATIONS, PRODUCTS
AND PLATFORMS.

FACTS + FIGURES

33

Global
Offices

30k⁺

Employees

100⁺

Fortune 500
Clients

GLOBAL R&D FOOTPRINT & SECURED FACILITIES

North America

Strategy, Consulting, Design,
Product Engineering, Data Services
· Localization

Location: Redmond, San Francisco,
Chicago and New York

Northeast Asia

Financial Service, IoT, Product Engineering ·
Language Services · Retail + Commerce · Data
Analytics

Location: China, Singapore, Japan, Taiwan

Western Europe

Content · Mobile Strategy · Localization ·
AI Enablement

Office: Barcelona, Dublin, Budapest,
Zurich.

India/Hyderabad

Product Engineering ·
Infrastructure/Ops Support ·
Data Integration · Retail +
Commerce ·



COMPREHENSIVE SERVICE OFFERING

pactera **EDGE**



ENGINEERING

- Enterprise Product Delivery
- Cloud Platform + DevOps
- Data Science + Data Platform
- AI Product Solutions
- Embedded Technology



DIGITALIZATION

- Transformation Consulting
- Digital Products
- Applications + Platforms
- Data + Analytics Modernization



GLOBALIZATION

- AI-Driven Language Services
- Product Internationalization
- Global Data Curation
- Global Websites
- Global Marketing Operations
- China Market Enablement



EMERGING TECHNOLOGIES

- Pact.AI
- IoT
- Blockchain
- Intelligent Automation + RPA
- Next Generation Experiences

**DIGITAL
AGENCY SERVICES
(BFM)**

**USER EXPERIENCE
DESIGN**

**CONTENT MANAGEMENT
PLATFORMS**

**ECOMMERCE
PLATFORMS**

**MARKETING
SERVICES**

**MARKETING
ANALYTICS + BI**

THE VALUE WE BRING

pactera **EDGE**



DOUBLE-AXIS VALUE PROPOSITION

RUN FASTER

Achieve new levels of
**performance to reduce
cost** and improve
operational efficiency.

RUN DIFFERENT

New **digital business
capabilities** to drive
relevance,
revenue & growth.

WORLDWIDE CLIENTS

	NORTH AMERICA & EUROPE	ASIA PACIFIC	CHINA
TECH	     	   	 
BFSI	   	      	   
TELE COMMUNICATION	  	 	   
MANUFACTURING & RETAIL	    	  	   
OTHERS	    	   	 

AGENDA

- PACTERA'S PoV
- CUSTOMER CHALLENGE
- CLOUD APPLICATION HAPPINESS JOURNEY
- CUSTOMERS SHOULD MOVE TO THE CLOUD AND SERVERLESS ENVIRONMENT
- APPLICATION MODERNIZATION WITH MICROSERVICES
- SERVERLESS (FUNCTIONS AS A SERVICE)
- AWS MICROSERVICES
- AWS CONTAINER AS A SERVICE
- AWS DEVOPS SERVICE OFFERINGS
- EVENT-DRIVEN DECOUPLE STATE FROM CODE USING MESSAGING
- CONCLUSION

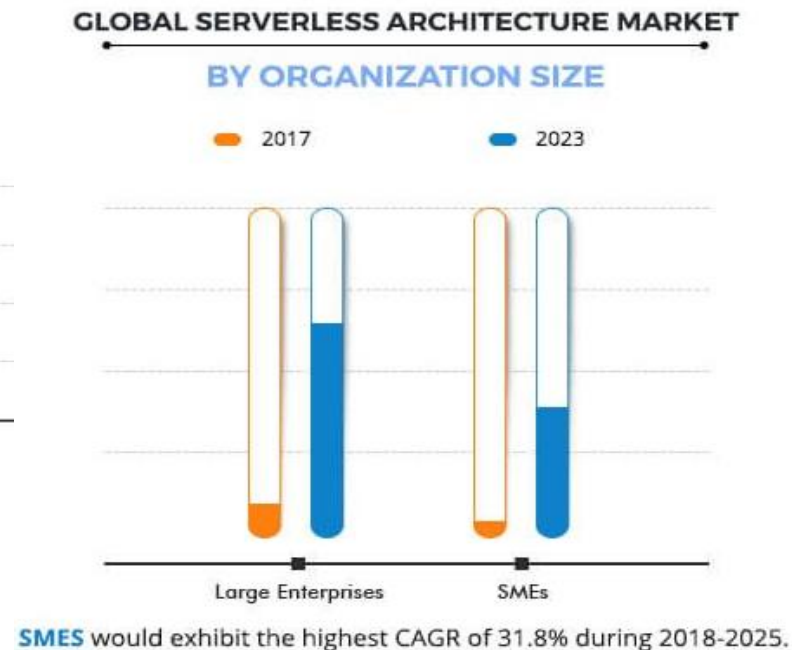
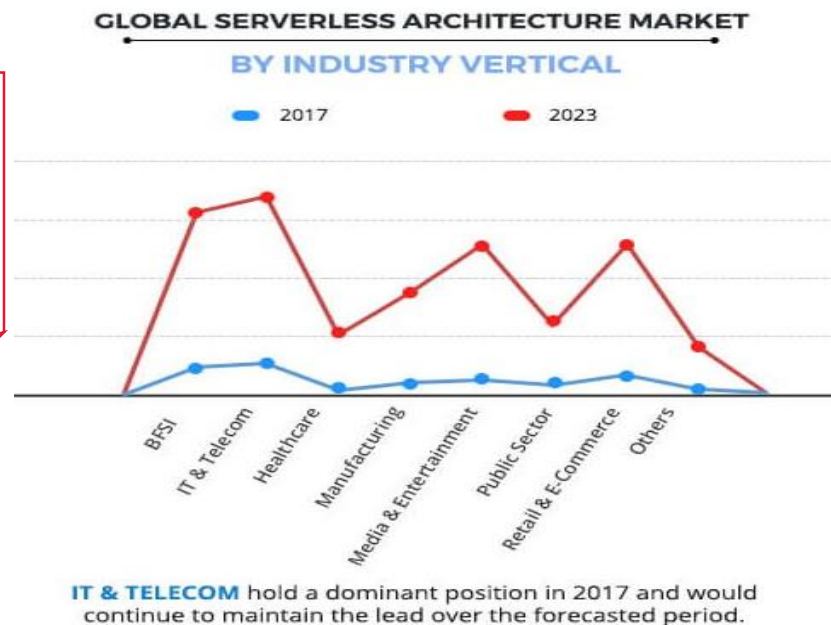


PACTERA's PoV

Serverless Computing

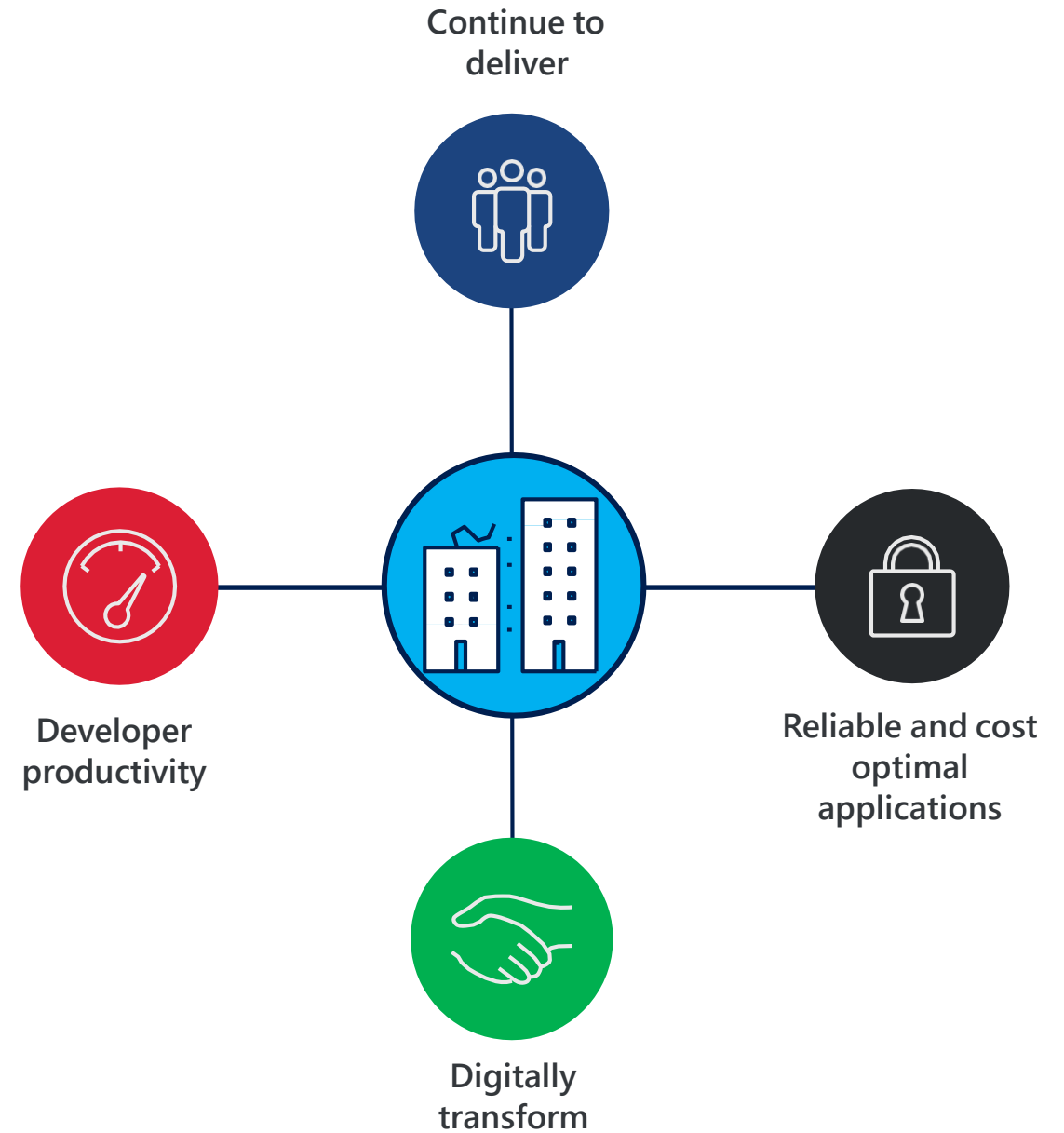
- Serverless computing is an emerging software architecture pattern that promises to eliminate the need for infrastructure provisioning and management.
- I&O leaders need to adopt an application-centric approach to serverless computing, managing APIs and SLAs, rather than physical infrastructures.

The global serverless architecture market size was valued at \$3,105.64 million in 2017 and is projected to reach \$21,988.07 million by 2025, registering a CAGR of 27.8% from 2018 to 2025

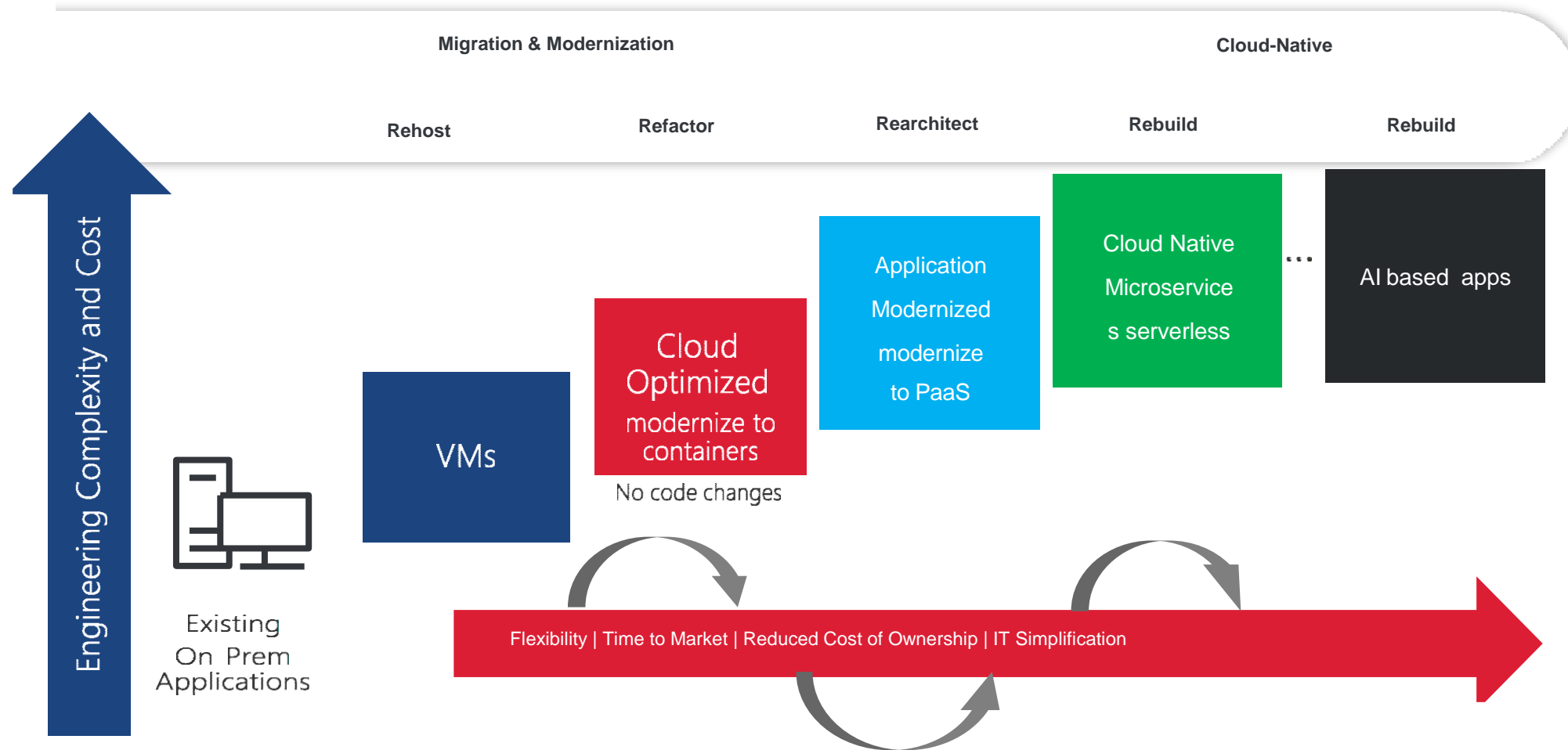


CUSTOMER CHALLENGE

- Developer productivity is essential for business agility
- Keeping data and applications reliable and cost optimal is critical
- Continue to rapidly deliver value to your customers
- Digitally transform your business to innovate and achieve happiness

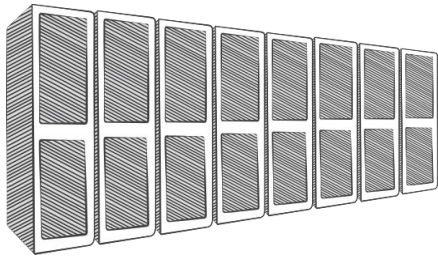


CLOUD APPLICATION HAPPINESS JOURNEY



WHY CUSTOMERS SHOULD MOVE TO THE CLOUD AND SERVERLESS ENVIRONMENT

On-Premises



- Hardware maintenance
- Scalability issue
- High CAPEX/OPEX
- Complex operations
- Performance Issues
- Real-time Monitoring and Reporting issues
- Capacity Planning and Management challenges
- Backup and DR problems
- Connectivity and latency challenges



- Modernize current IT asset base
- Quick deployments and updates
- Lower infrastructure costs
- No up-front commitment
- Pay-as-you-go
- Increase Business Agility
- Disaster Recovery
- Security
- Elasticity & High availability
- Increase speed & agility
- Reduce business risks
- Operational Resilience

Disadvantage:-

- Limitation of AWS EC2
- General Cloud Computing Issues

Serverless Environment PaaS



- No server management
- Effectively scale and reliable
- Building serverless applications
- Faster deployment
- Develop More Powerful Apps
- Reusable code
- Automated high availability
- increasing your agility and innovation
- Real-time analytics and processing
- Microservices, Container
- CI/CD model (Devops)

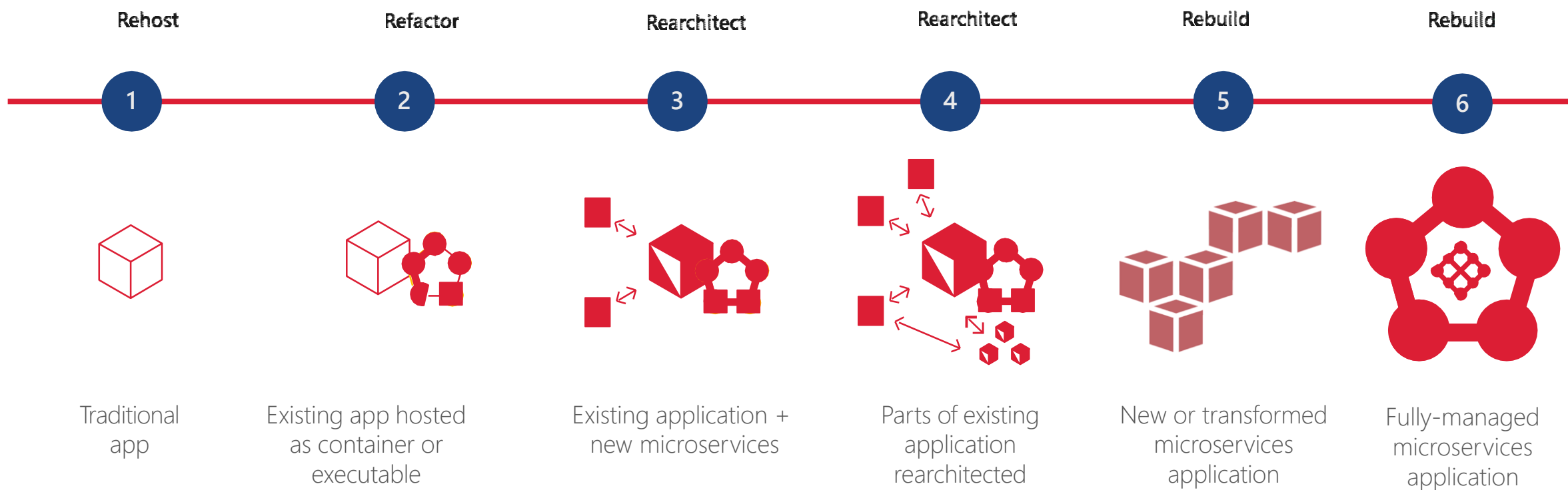
Disadvantage:-

- High Transaction cost
- Risk of Cutting-EDGE technologies

THE JOURNEY APPLICATION MODERNIZATION WITH MICROSERVICES

Migration & Modernization

Cloud-Native





SERVERLESS
(Functions as a Service)

WHY USE SERVERLESS (FAAS)?

Build and run applications without thinking about servers



Increase productivity

- Building serverless applications
- Reduced overhead
- Effectively scale and reliable
- Fully managed services

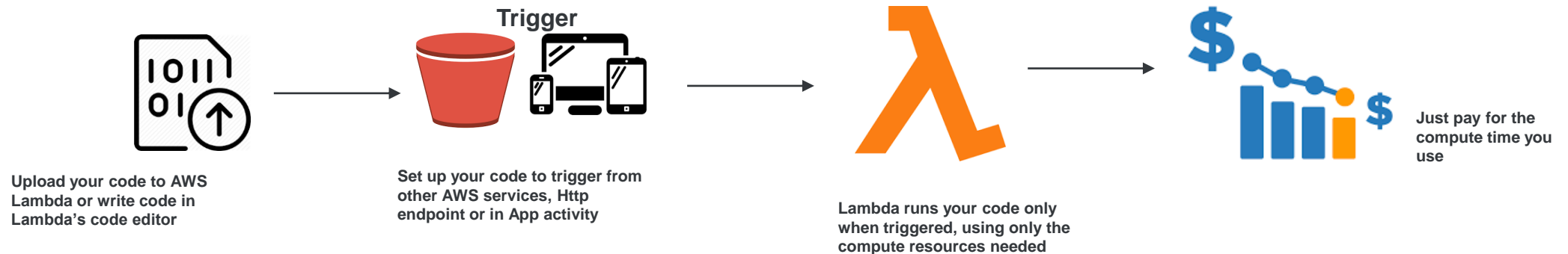
Harnessing cloud native features

- No server management
- Faster deployment
- Flexible scaling

Cost

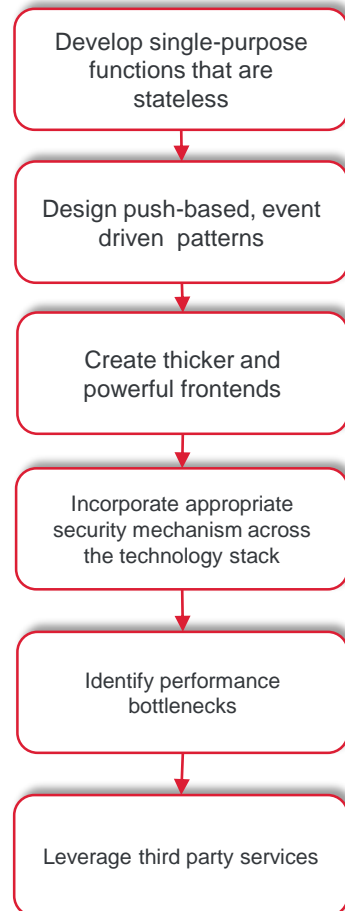
- Lower total cost of ownership
- Pay for consistent throughput or execution duration rather than by server unit

How its works



BENEFITS OF SERVERLESS PLATFORM & RECOMMENDED SERVERLESS ARCHITECTURAL PRINCIPLES

ARCHITECTURAL PRINCIPLES



Make your business more agile

- Make IT invisible
- Rationalize the applications
- Born in the cloud - Be free from the constraints of legacy IT.



Benefit from a serverless Architecture that replaces

- Frontloaded
- High-capex infrastructure with predictable costs
- Lower total cost of ownership, and optimal return on investment

BENEFITS



Create operational efficiency

- Become serverless
- Scale up
- Automate the core



Reinvent the business model

- Embrace a cloud model.
- Build intelligent
- Innovate with emerging technologies
- Serverless, Container , Microservice and Devops

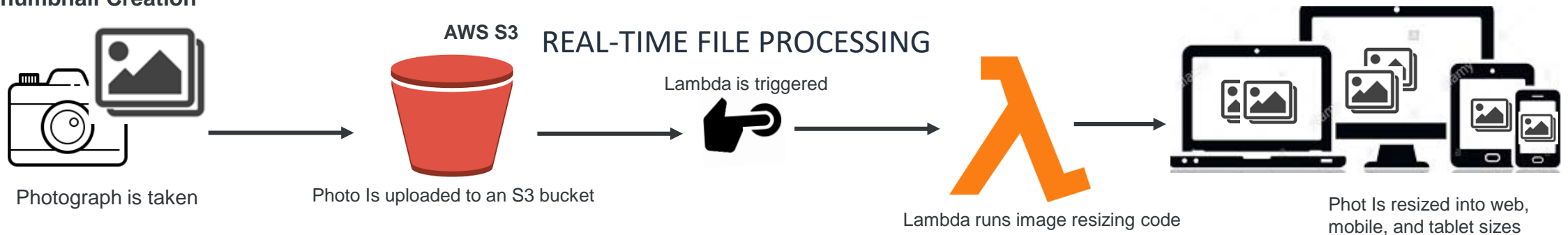
DATA PROCESSING WITH SERVERLESS

Build a variety of real-time data processing systems using AWS Lambda, Amazon Kinesis, Amazon S3, and Amazon DynamoDB.

REAL-TIME FILE PROCESSING

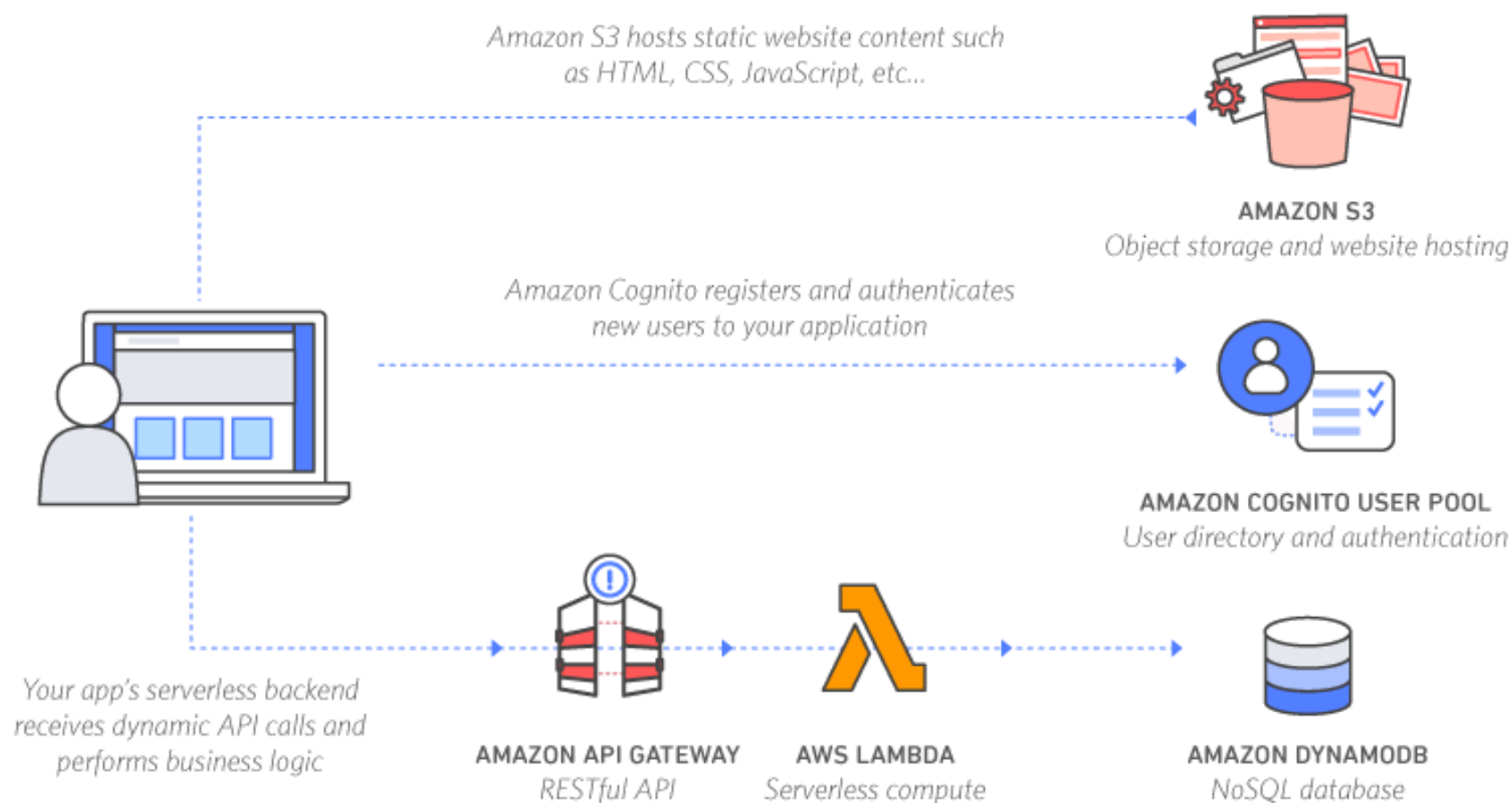
Amazon S3 to trigger AWS Lambda to process data immediately after an upload. For example, you can use Lambda to thumbnail images, transcode videos, index files, process logs, validate content, and aggregate and filter data in real-time

Image Thumbnail Creation



Reference architecture

BUILD A SERVERLESS APPLICATION



Benefits

- No server management
- Flexible scaling
- High availability
- No idle capacity
- Fully managed service:
- Less-Ops
- Pay for only execution time

CASE STUDY FOR SERVERLESS (LAMBDA)

The Challenge

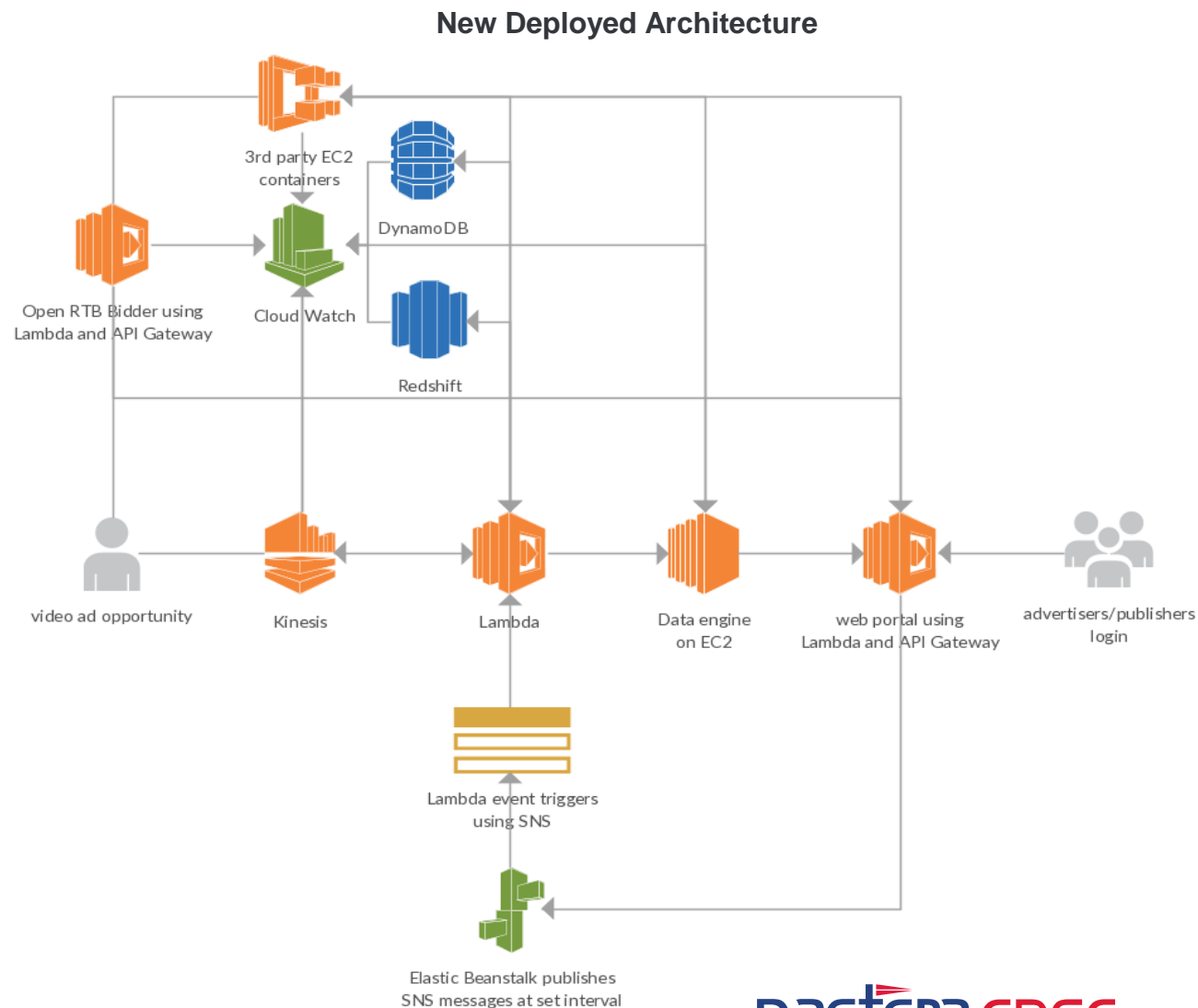
- **Customer** initially chose AWS because of its flexibility and scale.
- As **Customer** business grew, having developers manage a cluster of EC2 instances was becoming difficult despite using AWS Elastic Beanstalk to provision, manage, and scale the EC2 instances.
- There were always operations elements—instance type selection, scaling, deployment logic, and software configurations—for developers to manage.

The Benefits

- Using Lambda, **Customer** developers eliminate the need to understand or worry about infrastructure.
- Since the context the code is written in never changes, code does not need to be rewritten later as the system changes. This leads to productivity gains.
- Customer can now do with 2-3 engineers would usually take 8-10 engineers because code reusability becomes a growing performance advantage.
- Customer has grown revenue by 10x without hiring additional technical resources to manage volume, passing the cost savings to customers.

The Solution

- **Customer** now uses AWS Lambda to power the business logic for real-time ad bidding. The video player triggers a Lambda function through Amazon API Gateway.
- Lambda is also used to transcode video ads in real time.





AWS MICROSERVICES

WHY USE MICROSERVICES..?



Autonomous

- Developed
- Deployed
- Operated
- Scaled



Flexible Scaling

- Each service to be independently scaled
- Right-size infrastructure needs



Reusable Code

- Dividing software into small
- Well-defined modules



Specialized

- solving a specific problem
- developers contribute more code



Easy Deployment

- Microservices enable CI and CD



Resilience

- Degrading functionality
- Handle total service failure



Agility

- Shortens development cycle times
- Microservices foster an organization of small



Technological Freedom

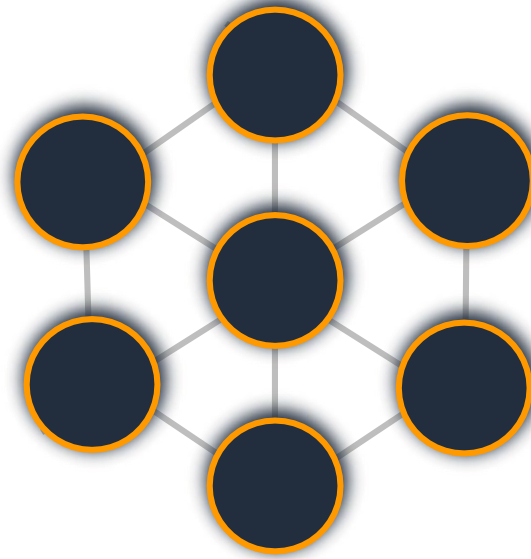
- One size fits all
- Choose the best tool

MONOLITHIC VS. MICROSERVICES



Monolith
Does everything

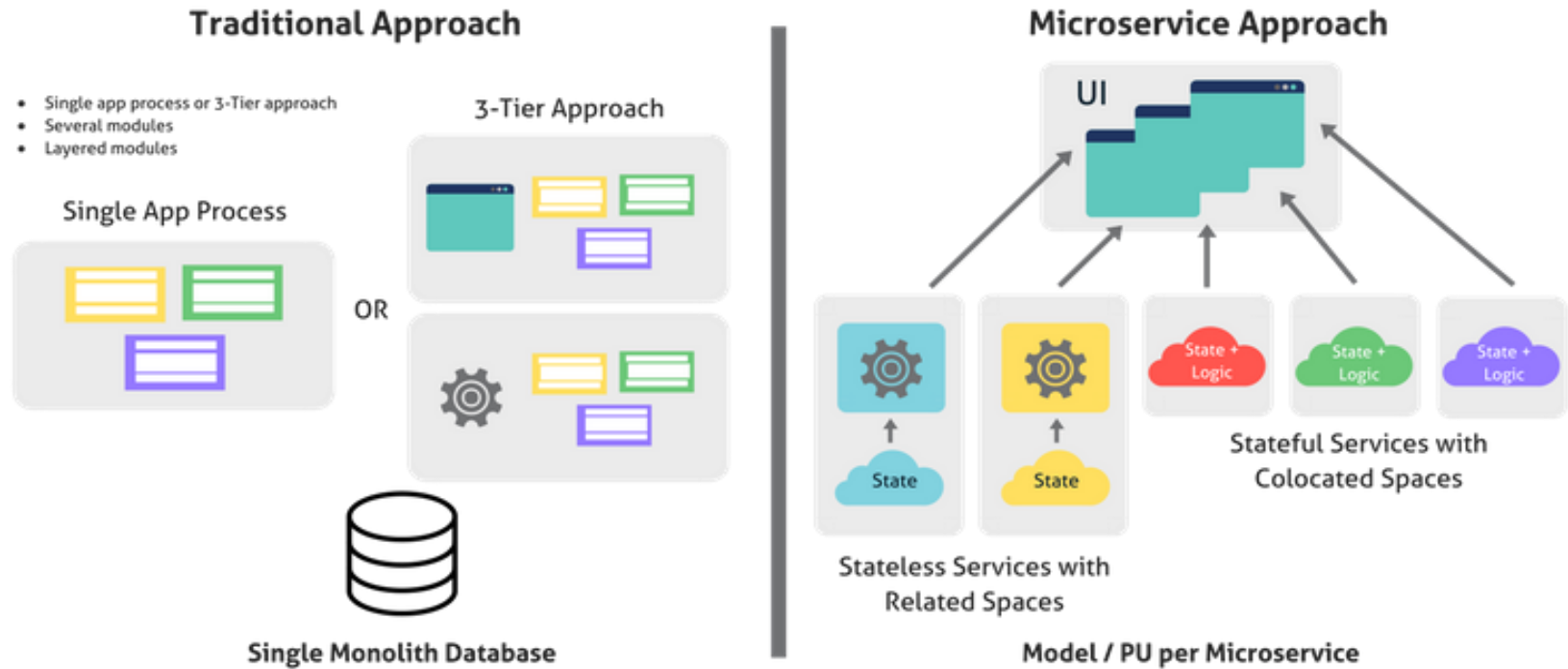
- Coordination overhead
- Poor modularity - No DRY
- High impact of change
- Poor scalability
- Long build time



Microservices
Does one thing

- Full ownership
- Full accountability
“DevOps”
- Focused innovation

MONOLITHIC VS MICROSERVICES ARCHITECTURE



Monolith Pros and Cons

Pros:

- Classic monolithic architectures provide better performance and strong coupling.
- Less reliance on third-party or other department's services.
- Full control of your application.

Cons:

- No agility for fast deployment or scalability.
- No simple way to make it highly available.
- Nearly impossible to isolate, compartmentalize or decouple system functionalities.

Microservices Pros and Cons

Pros:

- Allow you to scale up-and-out dynamically and on-demand.
- Agile methodology

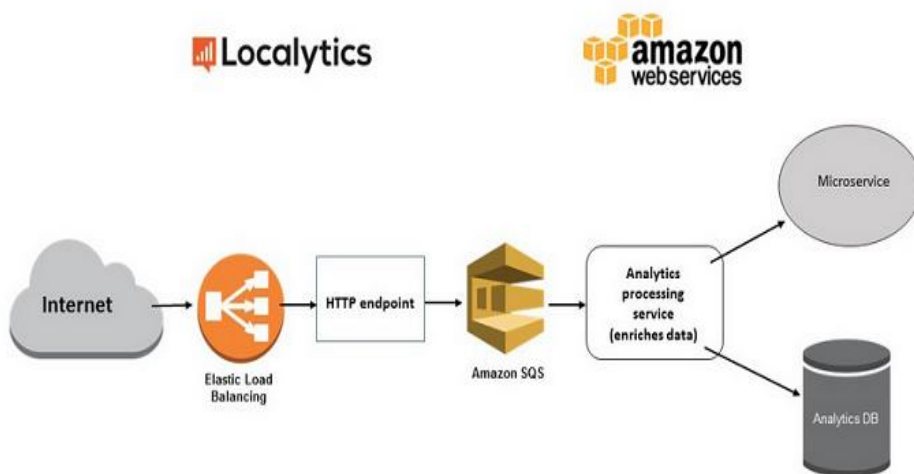
Cons:

- Granularity control
- Handle partial failure
- Multiple APIs
- Update multiple databases

MICROSERVICE- CASE STUDY

The Challenge

- Supports pipeline with billions of data points uploaded every day from different mobile applications running Localytics analytics software.
- Engineering team needed to access subsets of data for creating new services, but this led to additional capacity planning, utilization monitoring, and infrastructure management.
- Platform team wanted to enable self-service for engineering teams.

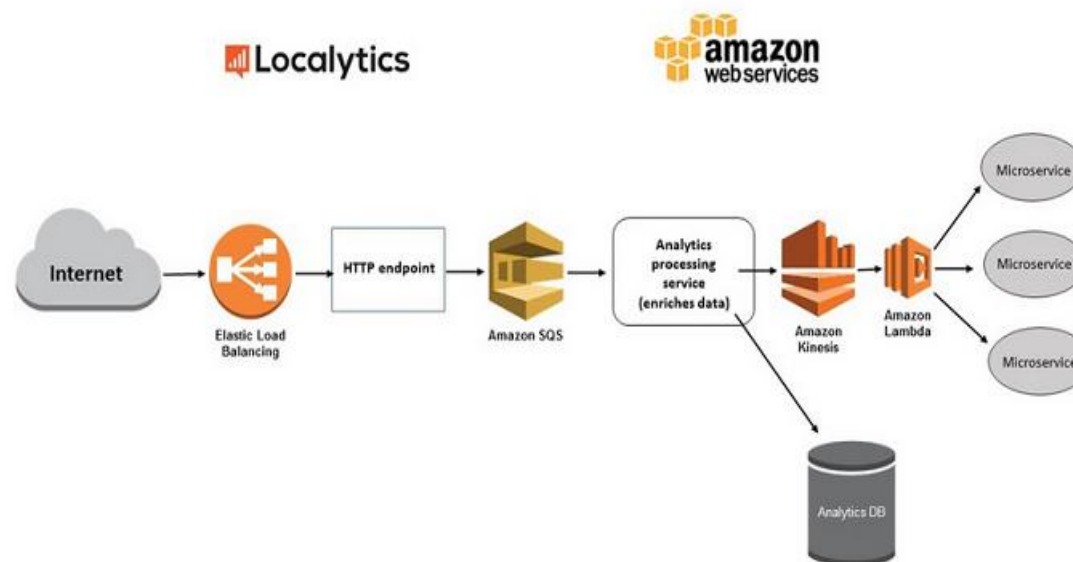


The Solution

- Uses AWS to send about 100 billion data points monthly through Elastic Load Balancing to Amazon Simple Queue Service, then to Amazon Elastic Compute Cloud, and finally into an Amazon Kinesis stream.
- For each new feature of the marketing software, a new microservice using AWS Lambda is created to access the Amazon Kinesis data stream. Each microservice can access the data stream in parallel with others.

The Benefits

- Decouples product engineering efforts from the platform analytics pipeline, enabling creation of new microservices to access data stream without the need to be bundled with the main analytics application.
- Eliminates the need to provision and manage infrastructure to run each microservice .
- Lambda automatically scales up and down with load, processing tens of billions of data points monthly.
- Speeds time to market for new customer services, since each feature is a new microservice that can run and scale independently of every other microservice.





AWS CONTAINER AS A SERVICE

WHY USE CONTAINER ..?

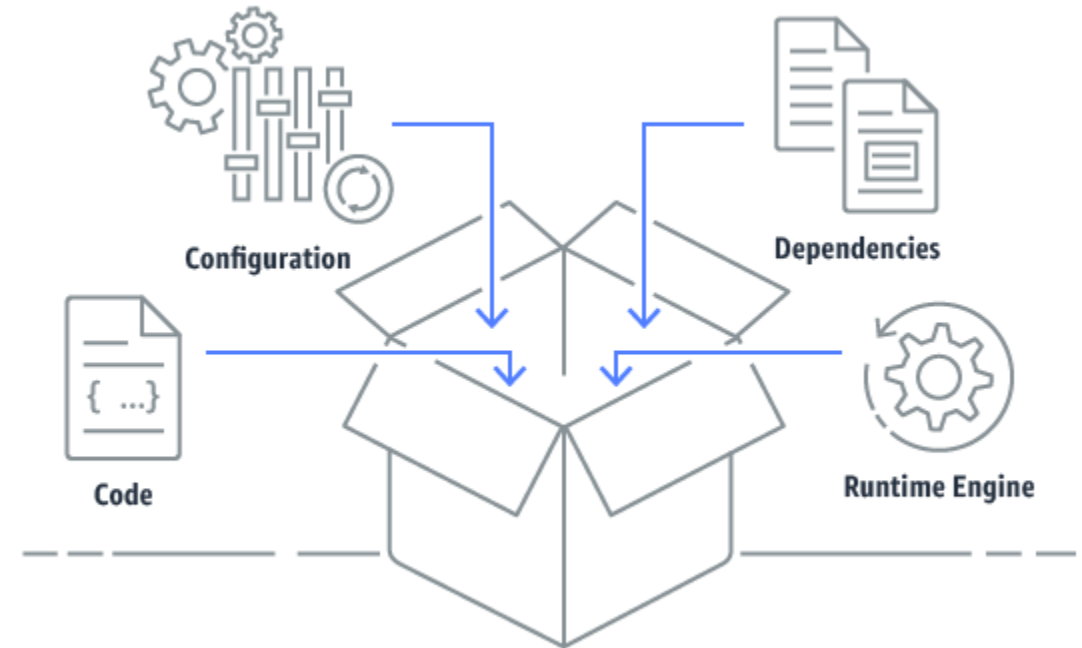


AMAZON ELASTIC CONTAINER AS A SERVICE



Run ECS containerized applications in production

- **Microservices**
 - Run microservices applications
 - Native integration (CI/CD) pipelines
- **Batch processing**
 - Batch workloads
 - Managed or custom schedulers
 - Reserved Instances
- **Application migration to the cloud**
 - Legacy enterprise applications
 - Containerized and easily migrated
- **Machine learning**
 - Easy to containerize ML models
 - Create ML models
 - Distributed services
 - Run anywhere
 - Improve resource utilization
 - Scale quickly



CONTAINER AS A SERVICE BENEFIT

Build and run applications without thinking about servers



Increase productivity

- Containers without servers
- Containerize everything
- Secure

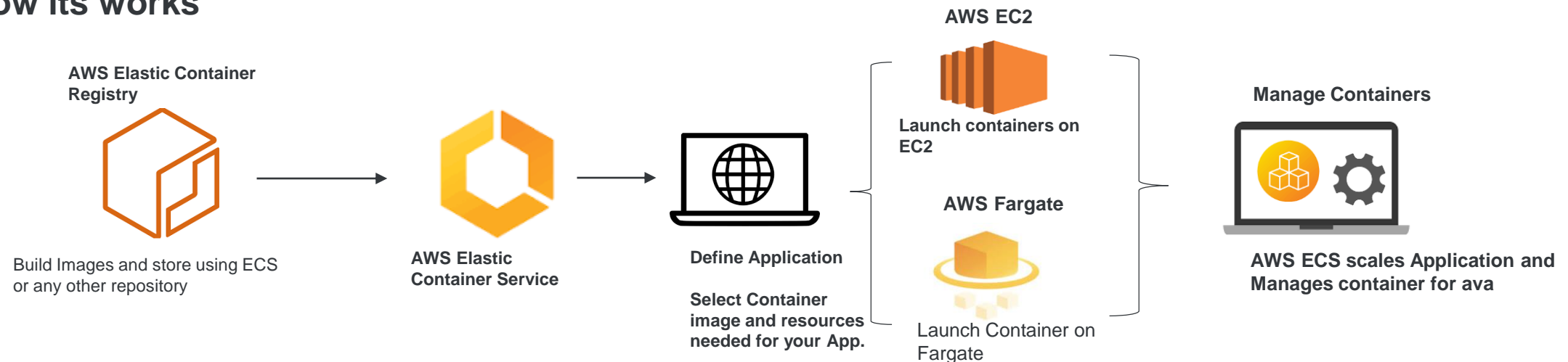
Harnessing cloud native features

- Faster deployment
- Flexible scaling
- Performance at scale

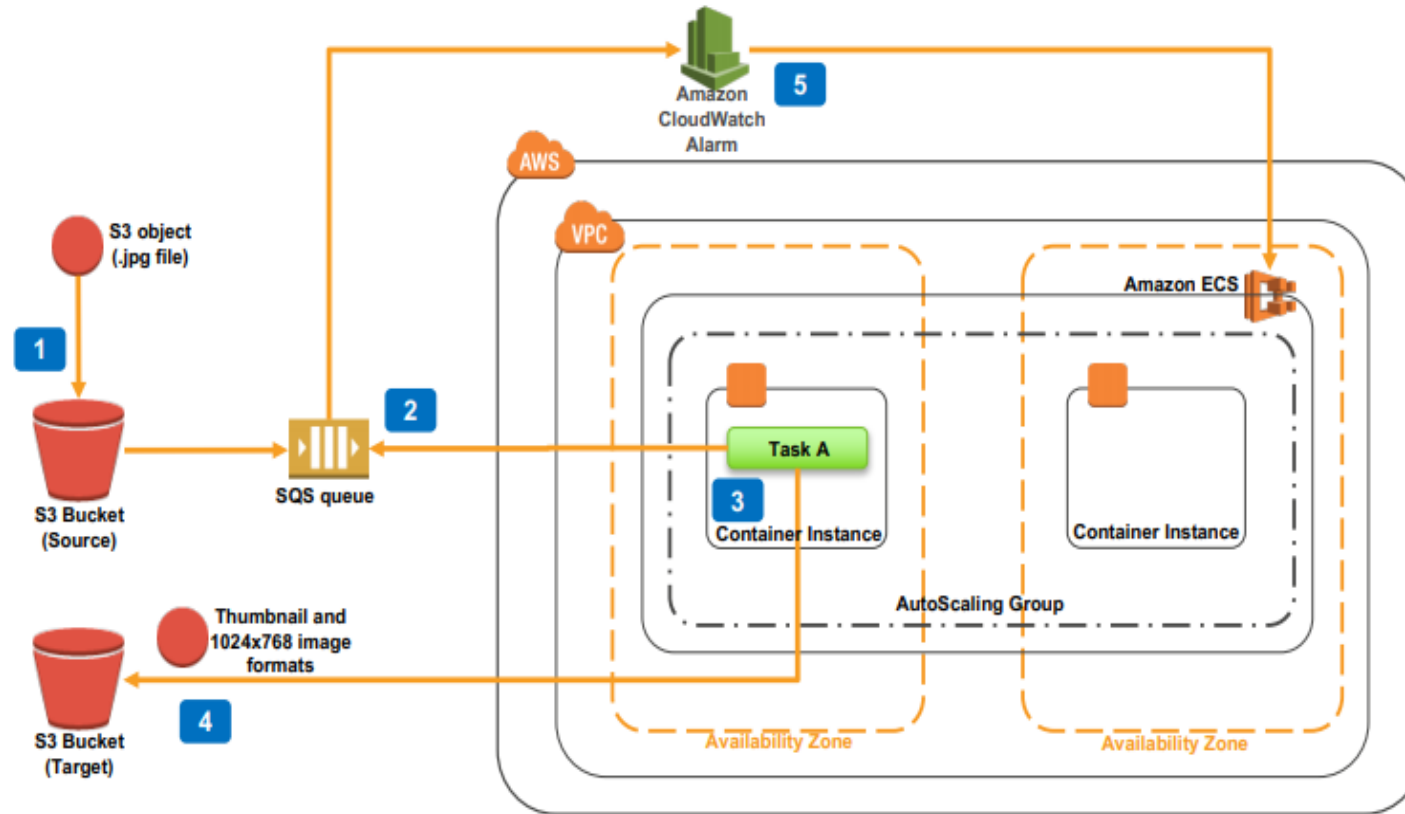
Cost

- Save up to 90% costs with spot instance

How its works



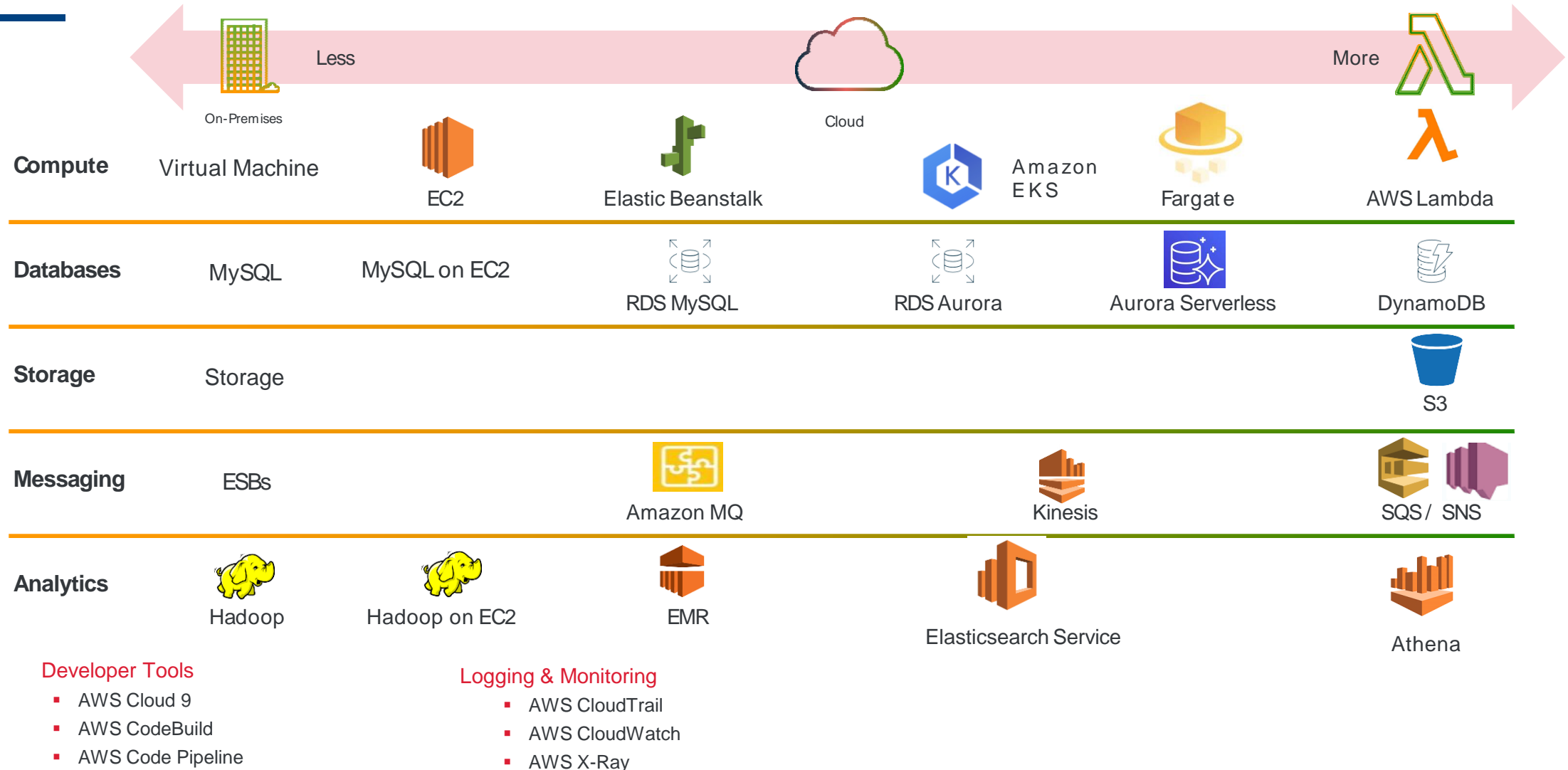
AMAZON ECS BATCH PROCESSING -SOLUTION



This diagram shows how to use Amazon S3, Amazon SQS, and Amazon ECS Build a batch processing framework to automate your batch jobs to build an automated batch processing framework.

1. An Amazon S3 object (a .jpg file) is inputted in the Amazon S3 Bucket that serves as the source bucket for batch jobs.
2. When the batch job is inputted into the Amazon S3 Bucket (Source), an event notification is sent to an Amazon SQS queue with the details of the object.
3. The Amazon ECS Task (batch worker) polls the Amazon SQS queue for new jobs (Amazon S3 objects).
4. If a job is available for processing, the task picks it up (a .jpg file) and processes it (generates a thumbnail and specific image format).
5. Cluster capacity (the number of tasks in the ECS service) is scaled up / down based on Amazon CloudWatch Alarms.
6. The metric used for scaling is the number of messages in the SQS queue.

AWS SERVICES FOR CONTAINERIZED





AWS DEVOPS SERVICE OFFERINGS

DEVOPS- CI/CD FRAMEWORK

How Amazon does DevOps



- Decompose for agility
 - *(microservices, 2 pizza teams)*
- Automate everything
- Standardized tools Belts and suspenders
 - *(governance, templates)*
- Infrastructure as code

AWS DEVOPS AS A SERVICE OFFERINGS



Continuous Integration

- Creates a modern application
- obstacles preventing
- version control



Continuous Delivery

- Rapid delivery
- Automate the Software Release Process
- Find and Address Bugs Quicker
- Deliver Updates Faster



Communication and Collaboration

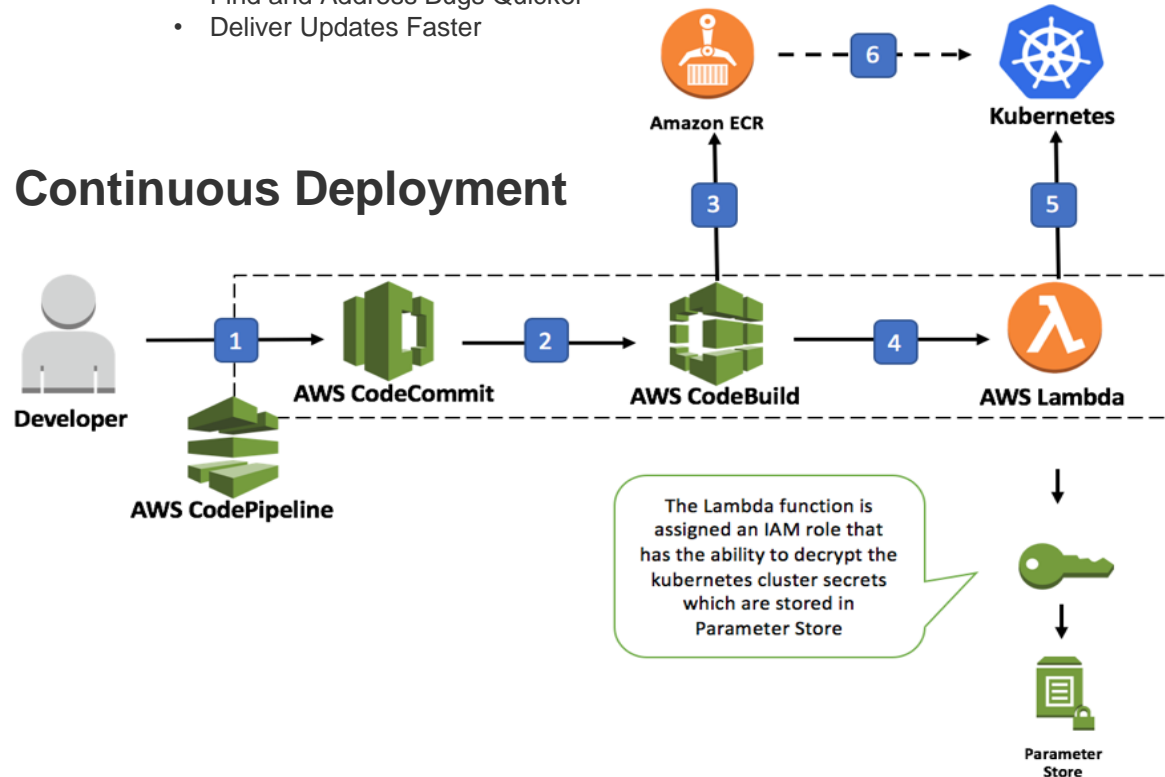
- Efficiency
- Intuitive
- Knowledge Sharing



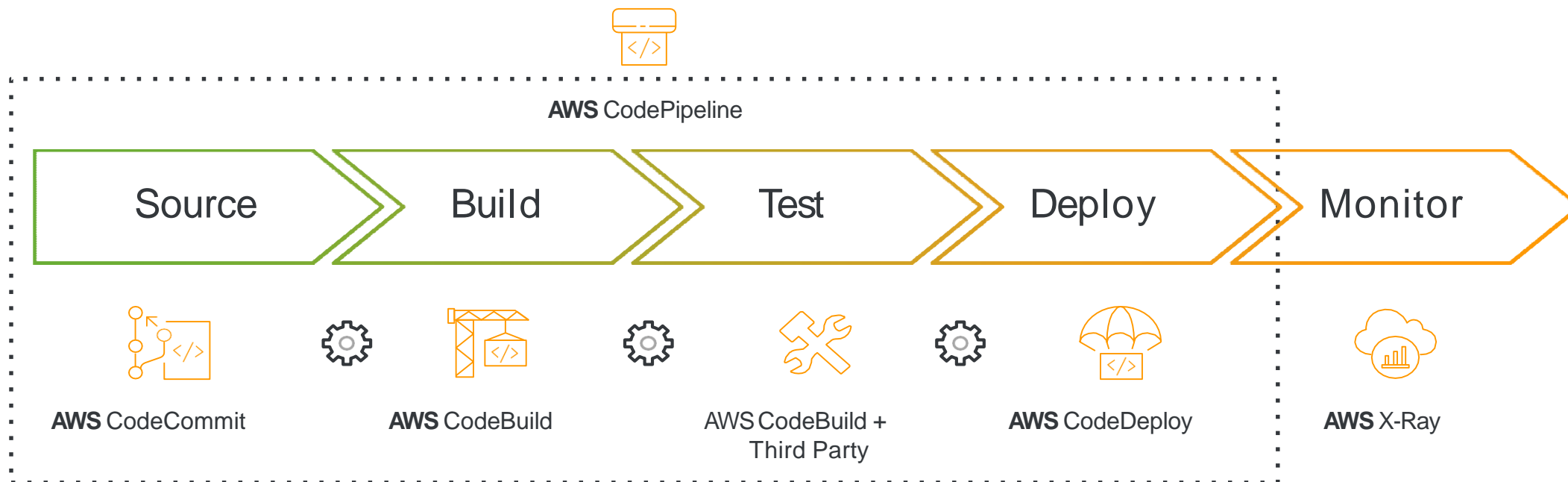
Infrastructure as Code

- Speed and simplicity
- Configuration consistency
- Minimization of risk
- Cost savings
- Increased efficiency in software development

An Example of Continuous Deployment



TOOLS FOR CI/CD



Software Release Workflows

AWS CodePipeline

- CI/CD services
- Fast & Reliable
- Release process models

Build and Test Code

AWS CodeBuild

- Fully managed build
- Compiles source code
- Scales continuously

Deployment Automation

AWS CodeDeploy

- Automate code deployment
- Rapidly release
- Easy to update applications

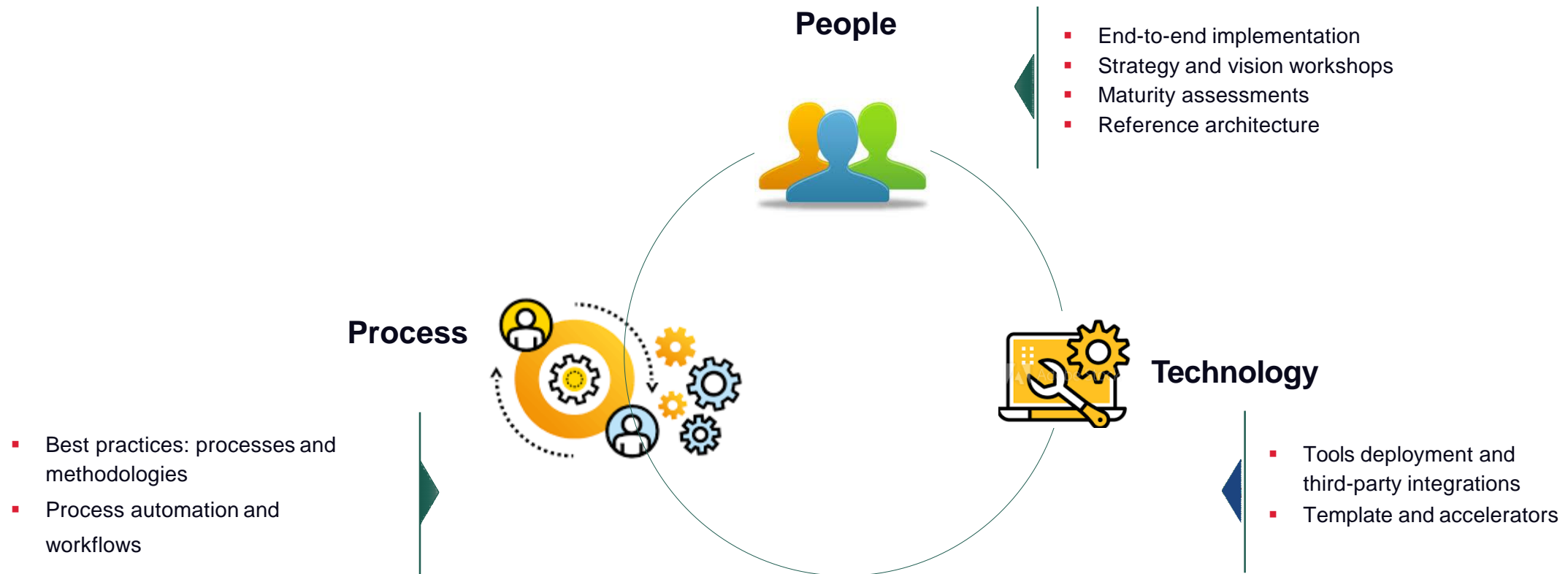
Unified CI/CD Projects

AWS CodeStar

- Quickly develop
- unified user interface
- easily manage software development
- releasing code faster.

DEVOPS AND AGILE METHODOLOGY

DevOps is a way to align people, processes, and technology to enable unified application delivery and increase business agility



CI/CD- SOLUTION ARCHITECTURE

The Challenge

- Instacart originally deployed every application using a homegrown deployment tool
- The tool required two to three hours of work from one or two developers every week to monitor and maintain, and was limited in functionality and features
- Company performs hundreds of deployments a day to clusters of different sizes, so needed a reliable way to deploy and to monitor those deployments

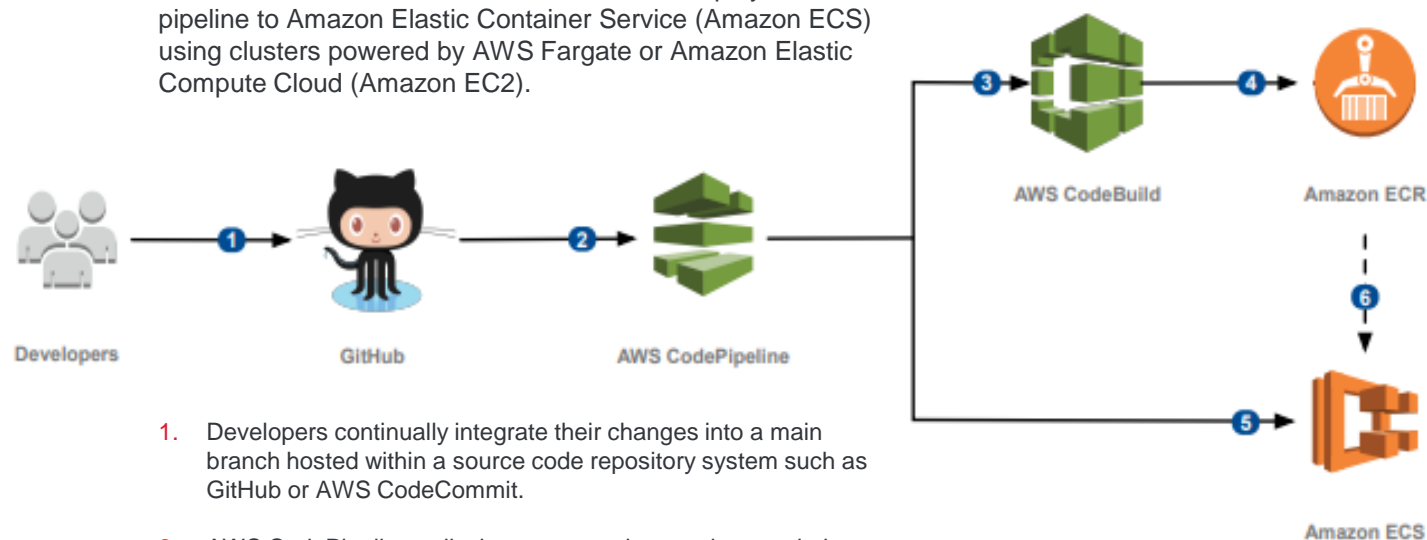
The Solution

- Started using AWS CodeDeploy to deploy all front-end and back-end services including consumer-facing websites, APIs, mobile apps, internal tools, messaging infrastructure, and processing systems
- CodeDeploy works with Instacart's existing continuous integration and delivery pipeline setup
- Engineers use the CodeDeploy console and CodeDeploy APIs to monitor the status of each deployment
- Uses CodeDeploy's deployment configurations options depending on the application being deployed and its SLA—rolling updates for consumer-facing web services and all-at-once or half-at-once updates for background job processing systems
- Uses CodeDeploy's lifecycle event hooks to automatically trigger scripts at different stages of each deployment, ensuring the proper configuration and libraries are automatically installed, verifying that applications are booted correctly, and notifying them if rollback updates have failed.

The Benefits

- Developers can focus on the core product and worry less about deployment operations
- Instacart team no longer needs to spend time and resources maintaining its own internal deployment tool

This diagram shows how to use AWS CodePipeline and AWS CodeBuild to build an automated continuous deployment pipeline to Amazon Elastic Container Service (Amazon ECS) using clusters powered by AWS Fargate or Amazon Elastic Compute Cloud (Amazon EC2).



1. Developers continually integrate their changes into a main branch hosted within a source code repository system such as GitHub or AWS CodeCommit.
2. AWS CodePipeline polls the source code repository and triggers an execution of the pipeline when a new revision is found.
3. AWS CodePipeline executes a build of the new revision in AWS CodeBuild which creates a Docker container image from the source code.
4. AWS CodeBuild pushes the newly built Docker container image tagged with the revision ID to an Amazon ECR repository.
5. AWS CodePipeline initiates an update of the Amazon ECS task definition and service with the new image location.
6. Amazon ECS fetches the new container from Amazon ECR and replaces the old tasks with the new one on the cluster.
7. The new revision of the service is now running on the cluster using the specified launch type: AWS Fargate or Amazon EC2.



AWS MESSAGING AS A SERVICE

EVENT-DRIVEN DECOUPLE STATE FROM CODE USING MESSAGING

Messaging



**Amazon Simple
Queue Service**

Queues

Simple
Fully-managed
Any volume



**Amazon Simple
Notification
Service**

Pub/sub

Simple
Fully-managed
Flexible



**Amazon
CloudWatch
Events**

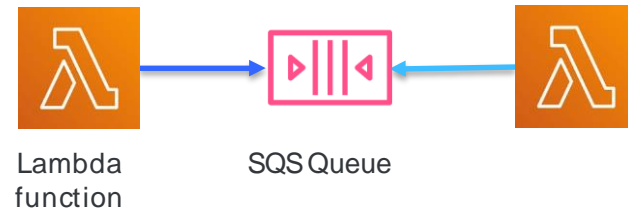
Synchronization

Rapid
Fully-managed
Real-time

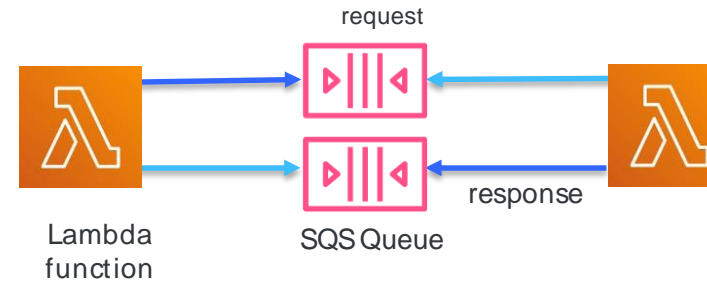
MICROSERVICES MESSAGING PATTERNS

Event-driven architectures

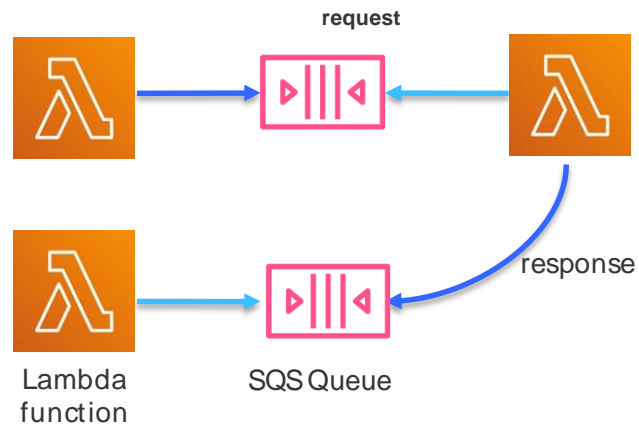
One-Way



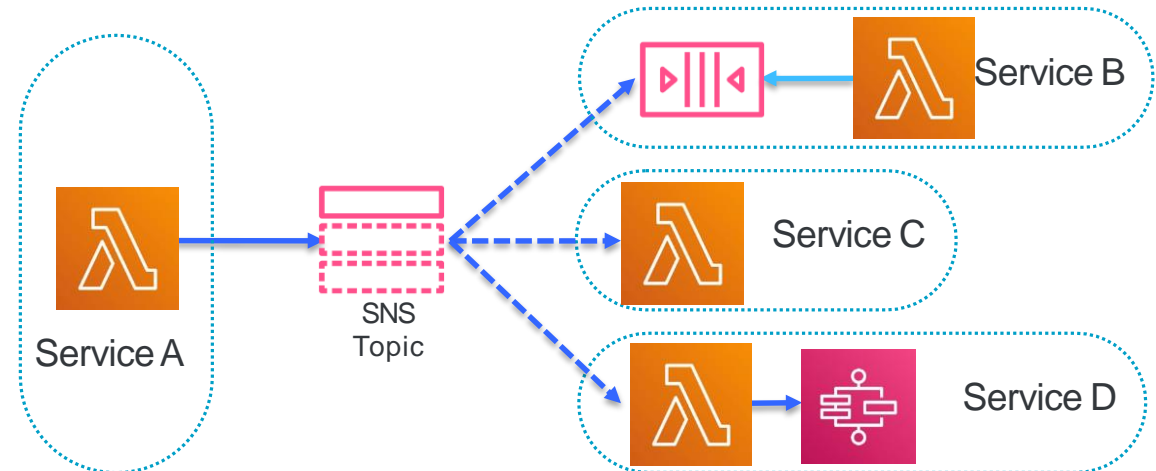
Request / Response



Return Address



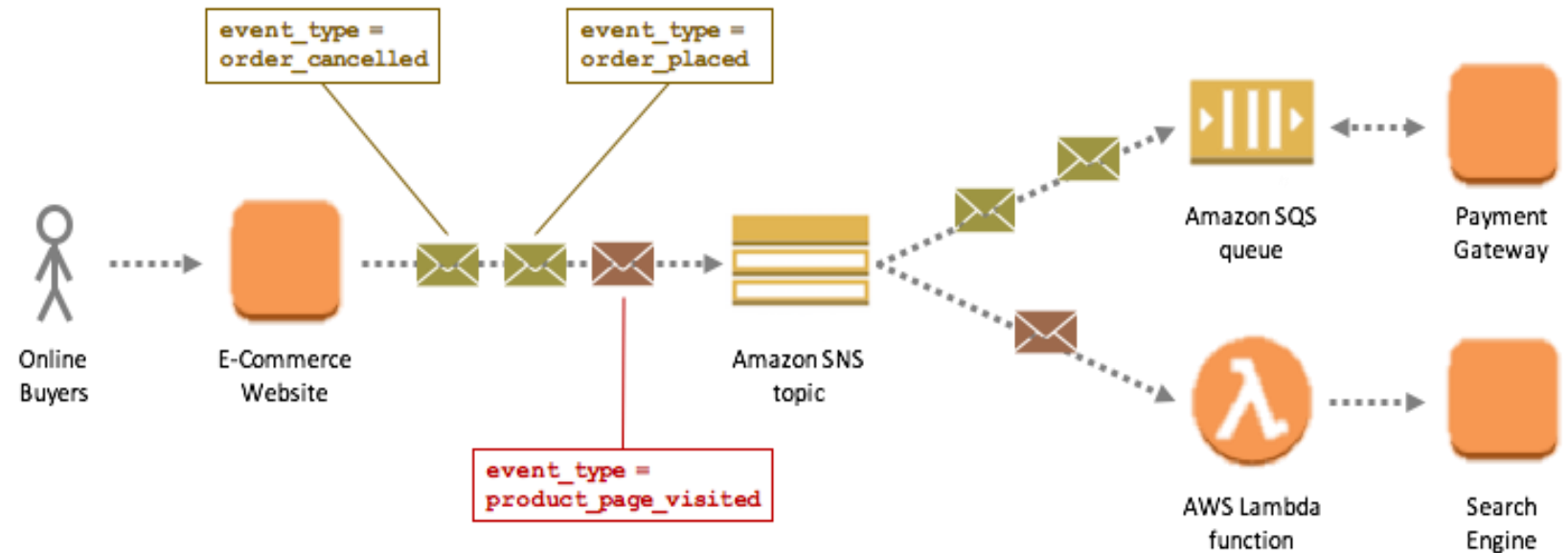
Publish / Subscribe



MESSAGE FILTERING SOLUTION ARCHITECTURE

Message filtering in action

- We recommend using message filtering and grouping subscribers into a single topic only when all the following is true.
- Subscribers are semantically related to each other
- Subscribers consume similar types of events
- Subscribers are supposed to share the same access permissions on the topic
- leverage the new message filtering capability
- SNS attempts to match the incoming message attributes








CONCLUSION

- Microservices architecture is a distributed design approach intended to overcome the limitations of traditional monolithic architectures.
- Microservices help to scale applications and organizations while improving cycle times. However, they also come with a couple of challenges that might add additional architectural complexity and operational burden.
- AWS offers a large portfolio of managed services that can help product teams build microservices architectures and minimize architectural and operational complexity. This whitepaper guides you through the relevant AWS services and how to implement typical patterns, such as service discovery or event sourcing, natively with AWS services.

KEY CUSTOMERS

E











D






G







BFM











THANK YOU