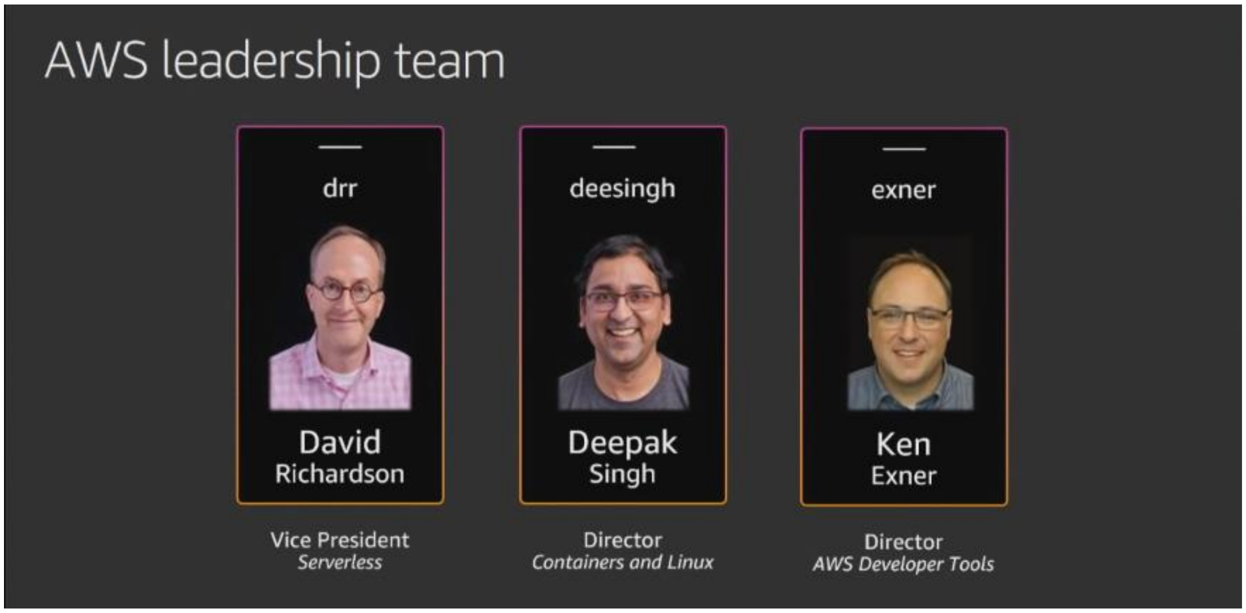
SRV325

Using DevOps, Microservices, and Serverless to Accelerate Innovation: A Leadership Discussion

re:Invent

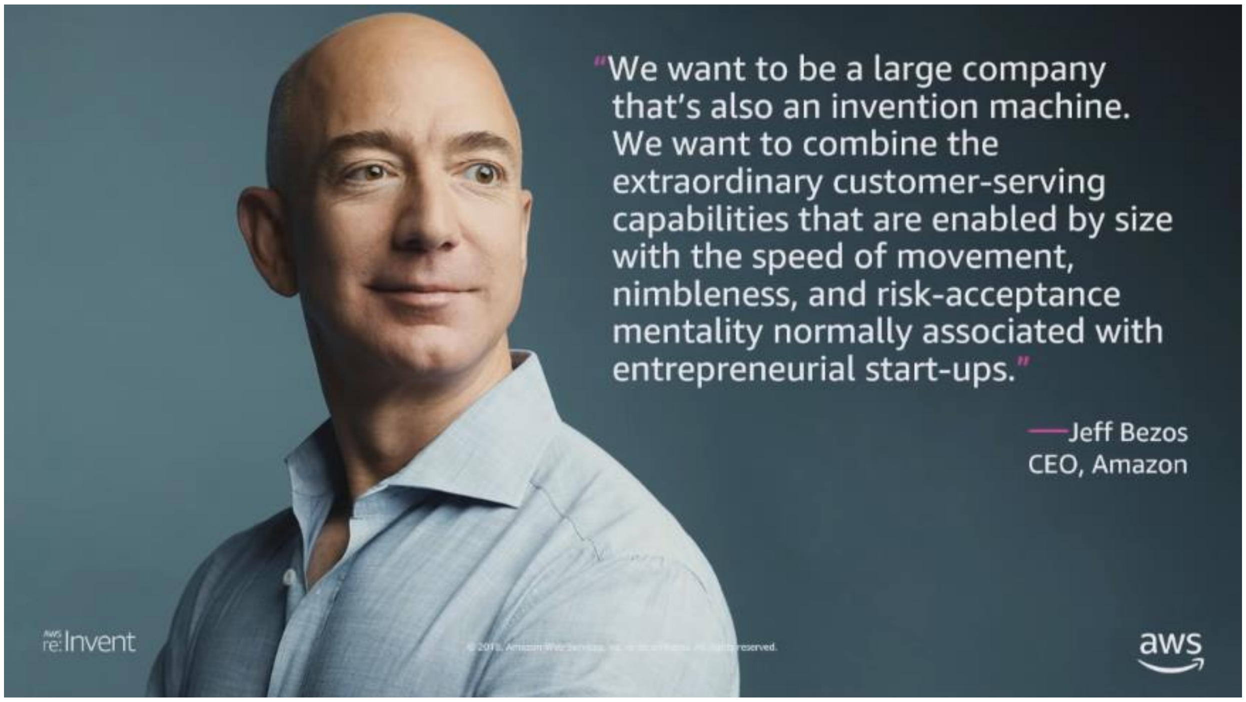
aws

In this session, learn how AWS can help you innovate faster with DevOps, microservices, and serverless. Join us for a rare and intimate discussion with AWS senior leaders: David Richardson, VP of Serverless, Ken Exner, director of AWS Developer Tools, and Deepak Singh, director of Compute Services, Containers, and Linux. Hear them share development best practices and discuss key learnings from building modern applications at Amazon.com. Also, learn how developers can leverage containers, AWS Lambda, and developer tools to build and run production applications in the cloud. Complete Title: AWS re:Invent 2018: Leadership Session: Using DevOps, Microservices, & Serverless to Accelerate Innovation (SRV325).



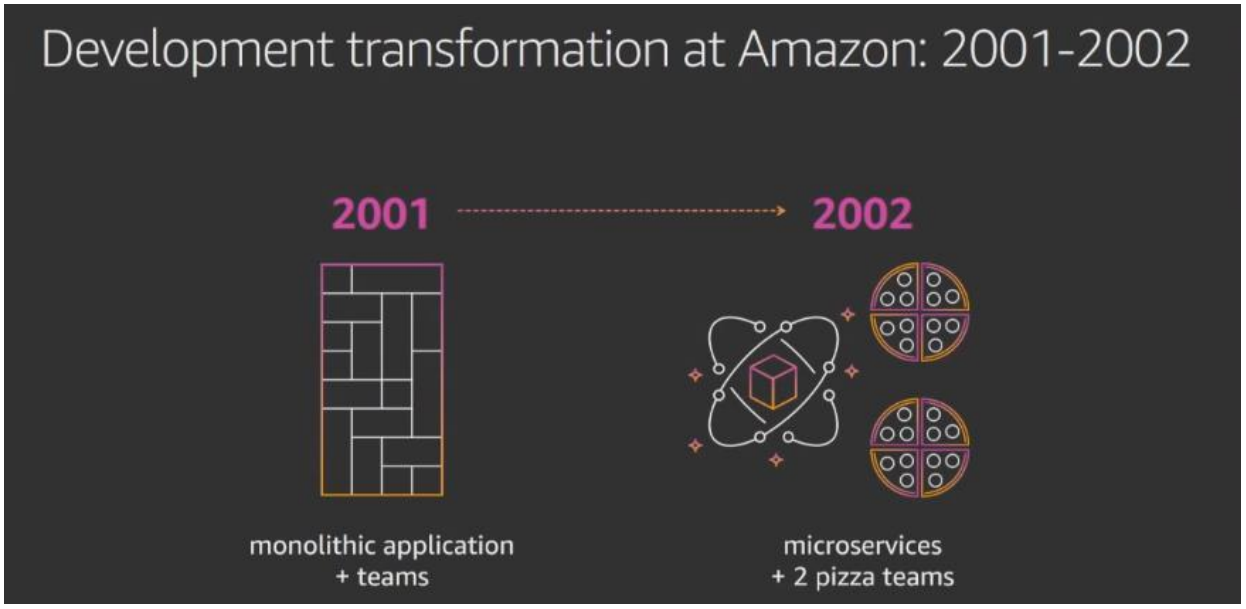
AWS leadership team

drr
David Richardson
Vice President *Serverless*
deesingh
Deepak Singh
Director *Containers and Linux*
exner
Ken Exner
Director *AWS Developer Tools*

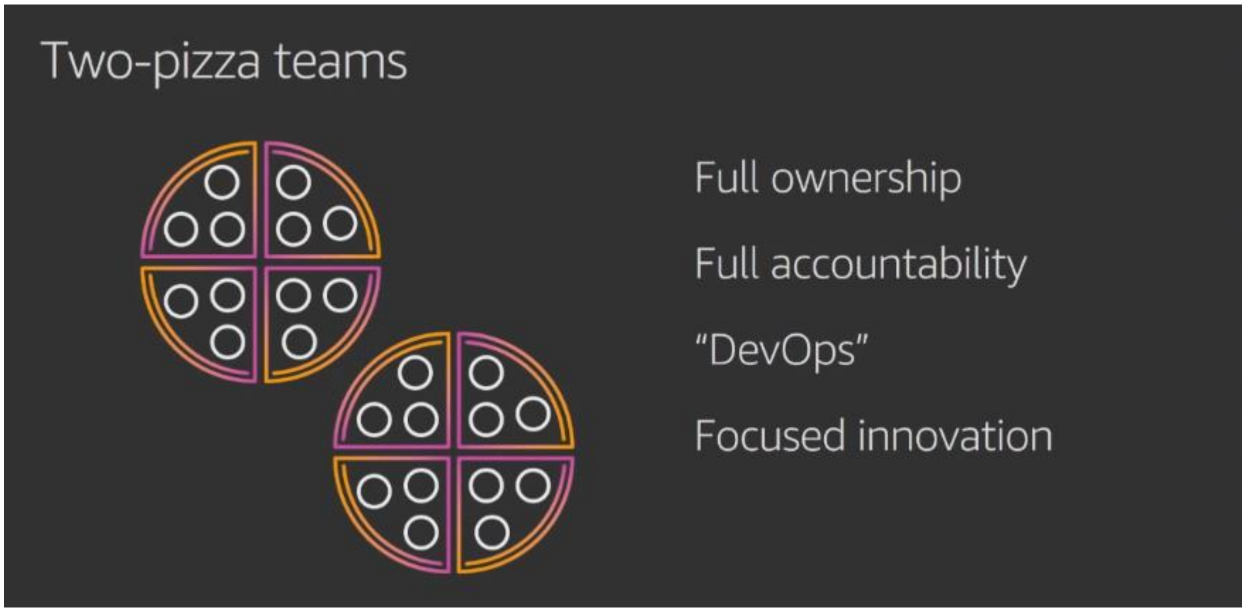


"We want to be a large company that's also an invention machine. We want to combine the extraordinary customer-serving capabilities that are enabled by size with the speed of movement, nimbleness, and risk-acceptance mentality normally associated with entrepreneurial start-ups."

—Jeff Bezos
CEO, Amazon



Development transformation at Amazon: 2001-2002



Two-pizza teams

Full ownership

Full accountability

“DevOps”

Focused innovation

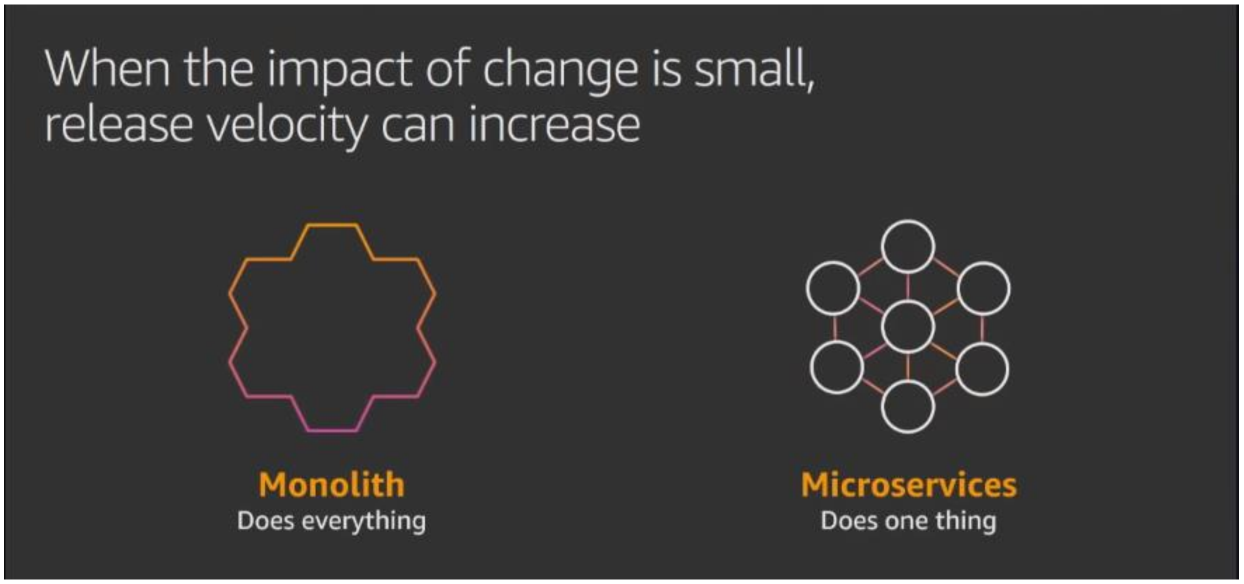
What changes have to be made in this new world?

Architectural patterns

Operational model

Software delivery

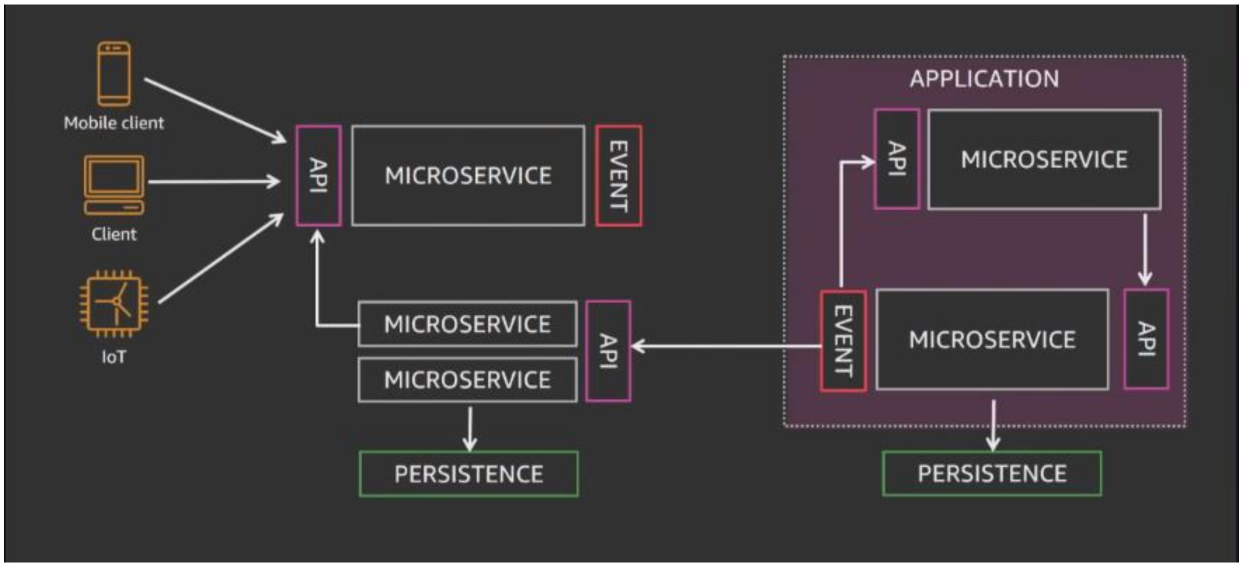
Changes to the architectural patterns



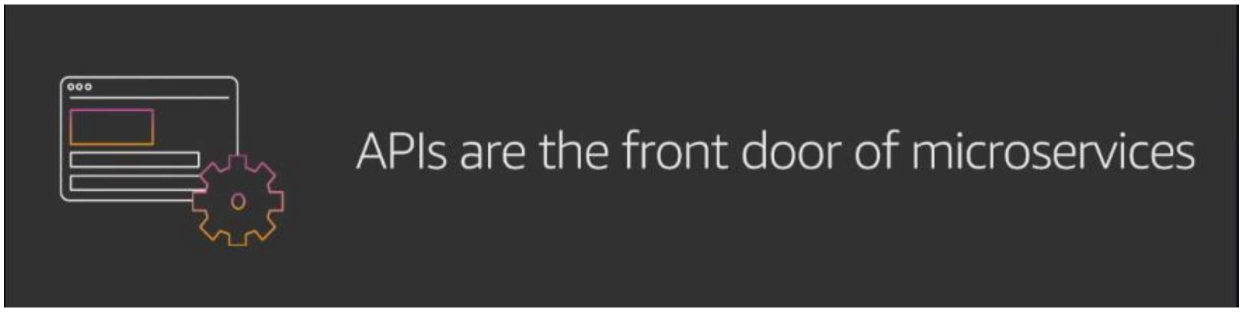
When the impact of change is small, release velocity can increase

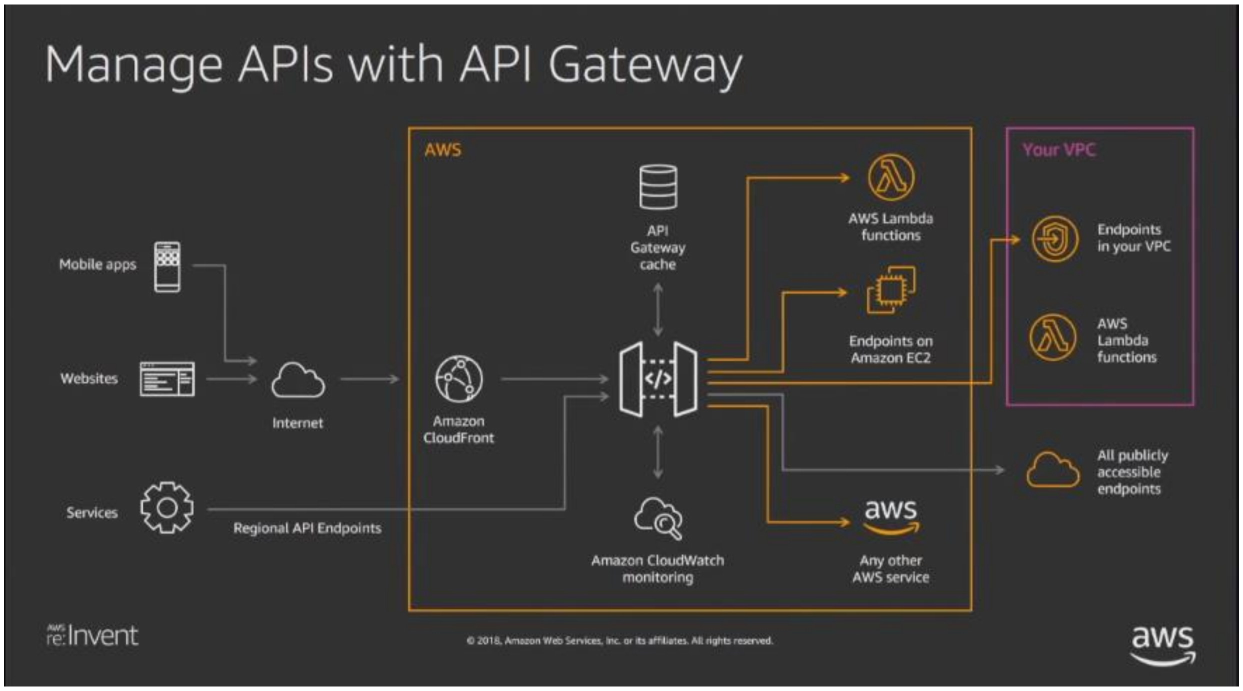
Monolith
Does everything

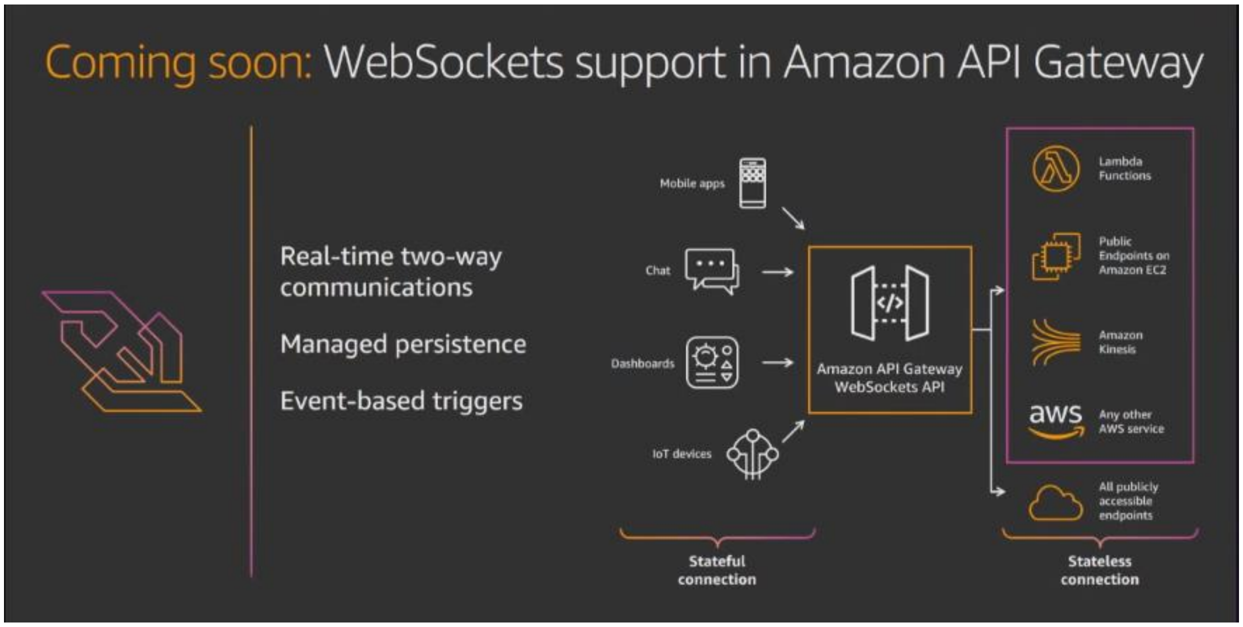
Microservices
Does one thing



There are 3 main patterns for building microservices architectures


APIs are the front door of microservices


Manage APIs with API Gateway

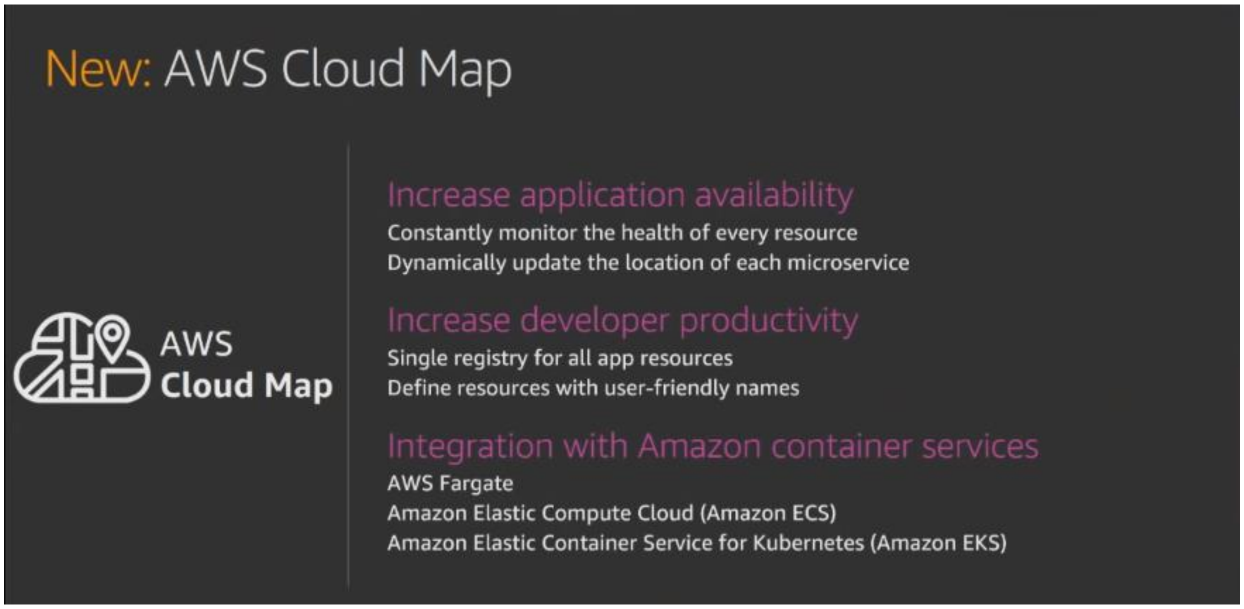


Coming soon: WebSockets support in Amazon API Gateway

Real-time two-way communications

Managed persistence

Event-based triggers



New: AWS Cloud Map

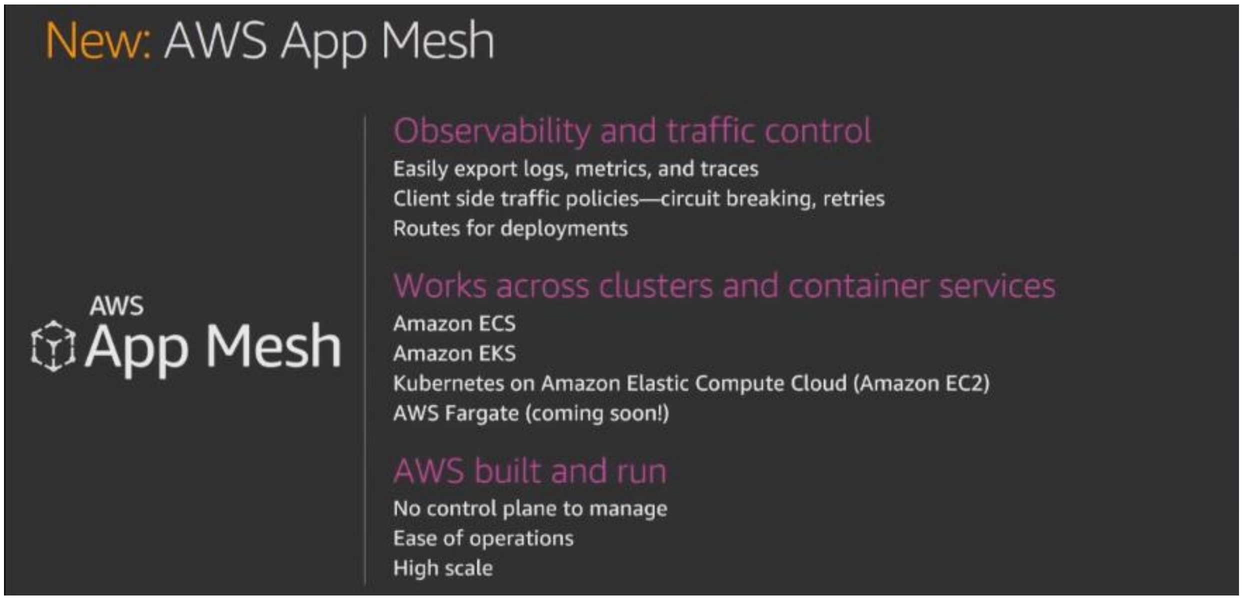
AWS Cloud Map

Increase application availability
Constantly monitor the health of every resource
Dynamically update the location of each microservice

Increase developer productivity
Single registry for all app resources
Define resources with user-friendly names

Integration with Amazon container services
AWS Fargate
Amazon Elastic Compute Cloud (Amazon ECS)
Amazon Elastic Container Service for Kubernetes (Amazon EKS)

When you build microservices with lots of APIs, you might need something to maintain that distributed environment



New: AWS App Mesh

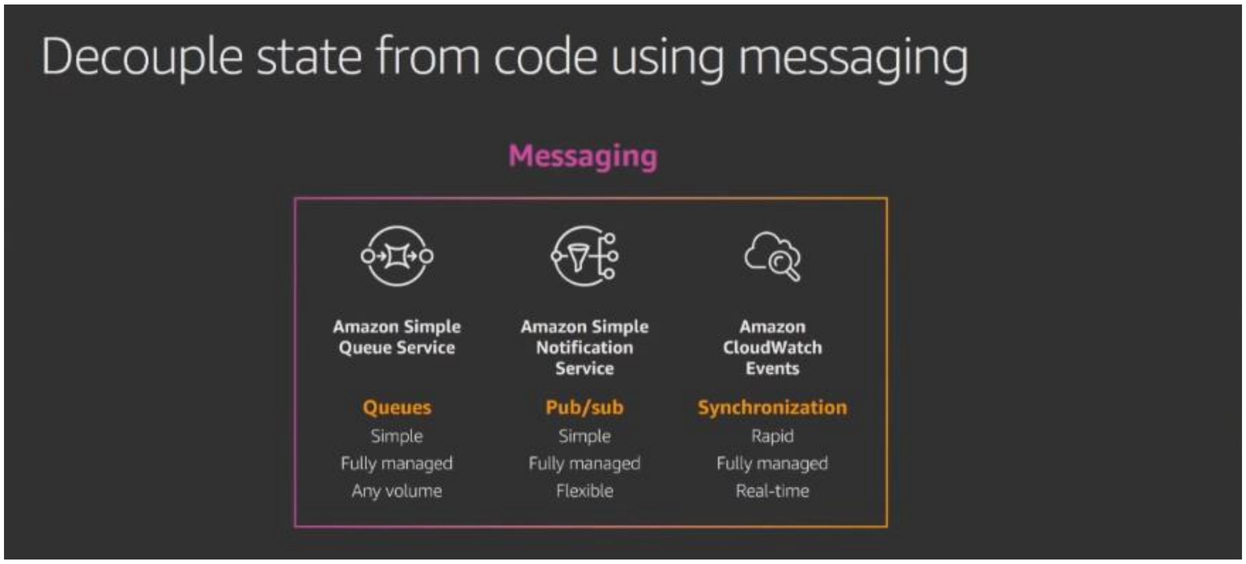
AWS App Mesh

Observability and traffic control
Easily export logs, metrics, and traces
Client side traffic policies—circuit breaking, retries
Routes for deployments

Works across clusters and container services
Amazon ECS
Amazon EKS
Kubernetes on Amazon Elastic Compute Cloud (Amazon EC2)
AWS Fargate (coming soon!)

AWS built and run
No control plane to manage
Ease of operations
High scale

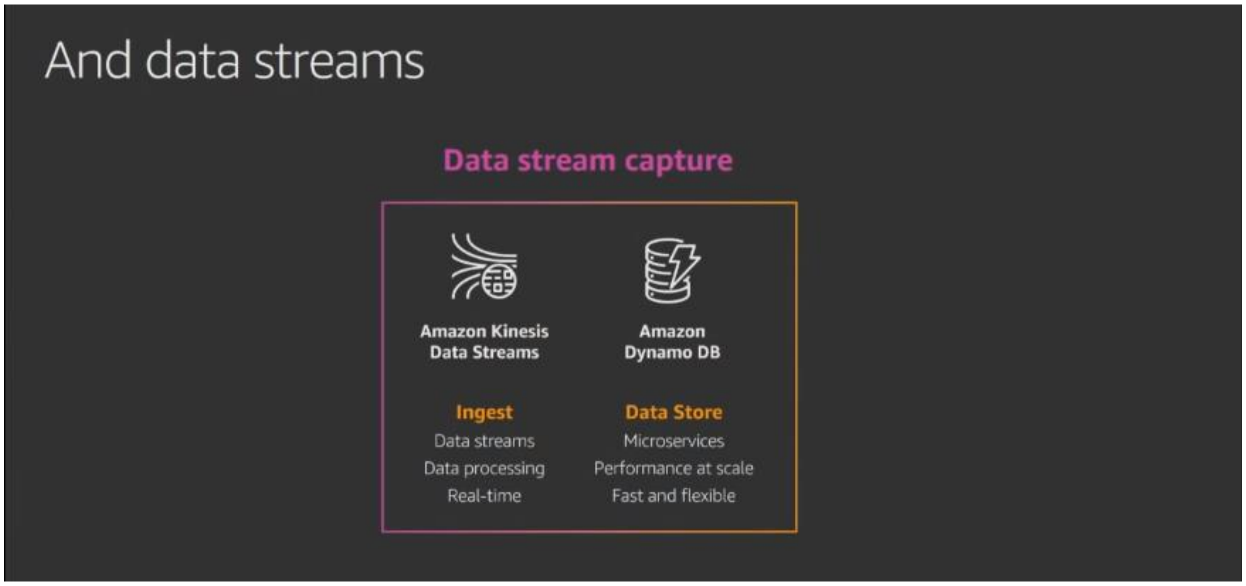
Event-driven architectures



Decouple state from code using messaging

Messaging

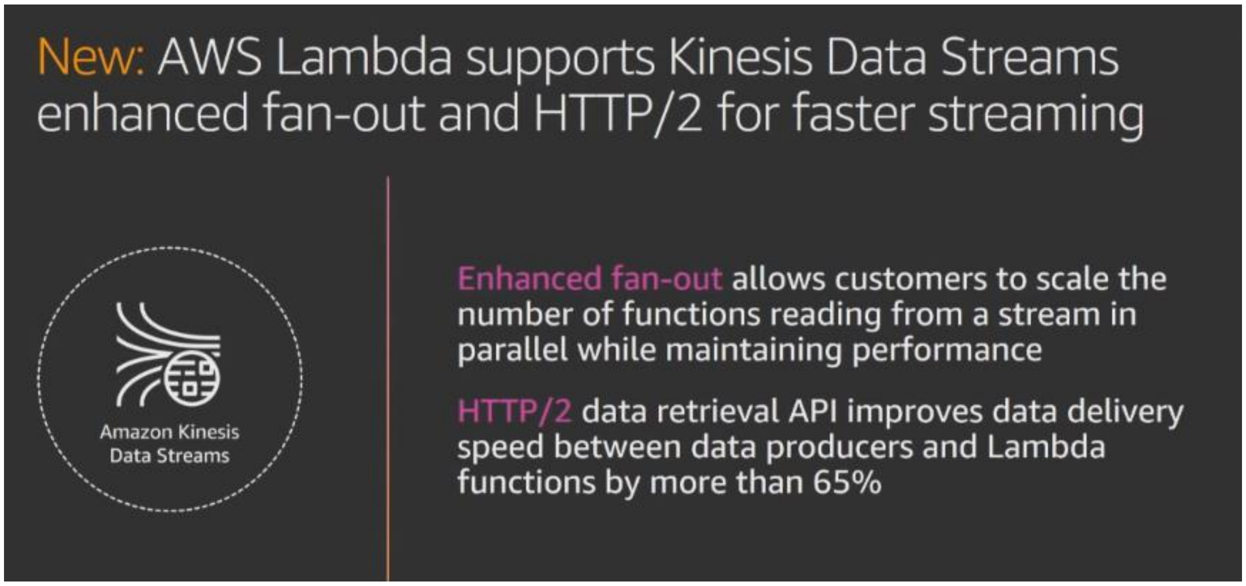
Amazon Simple Queue Service	Amazon Simple Notification Service	Amazon CloudWatch Events
Queues	**Pub/sub**	**Synchronization**
Simple	Simple	Rapid
Fully managed	Fully managed	Fully managed
Any volume	Flexible	Real-time



And data streams

Data stream capture

Amazon Kinesis Data Streams	Amazon Dynamo DB
Ingest	**Data Store**
Data streams	Microservices
Data processing	Performance at scale
Real-time	Fast and flexible

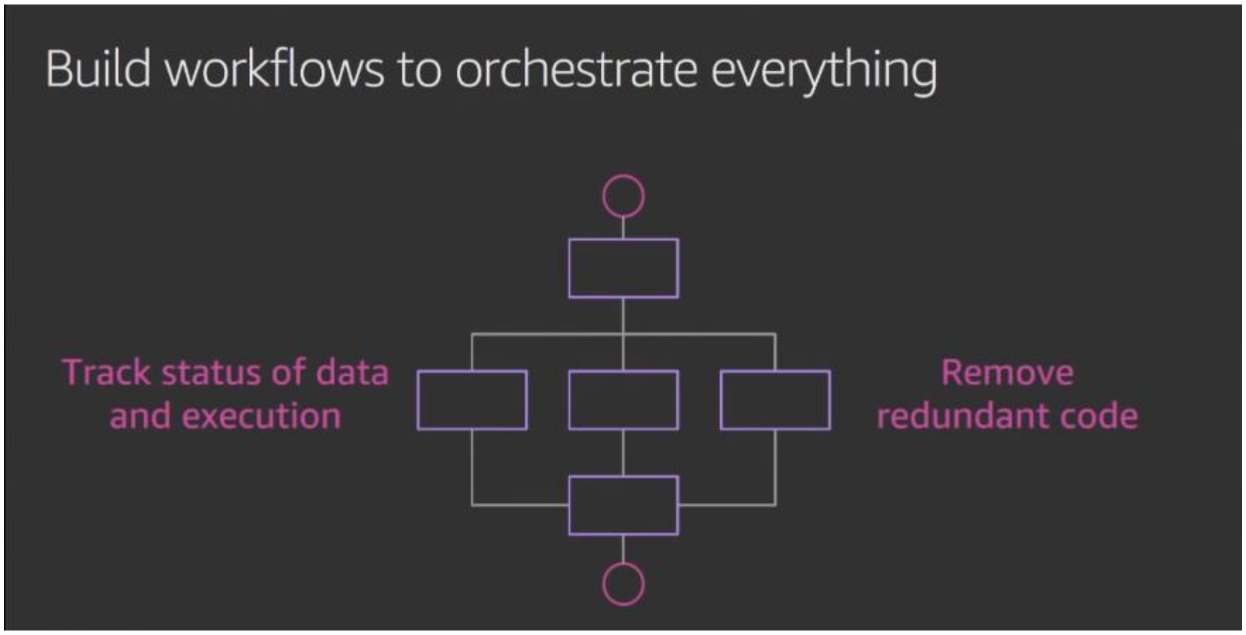


New: AWS Lambda supports Kinesis Data Streams enhanced fan-out and HTTP/2 for faster streaming

Amazon Kinesis Data Streams

Enhanced fan-out allows customers to scale the number of functions reading from a stream in parallel while maintaining performance

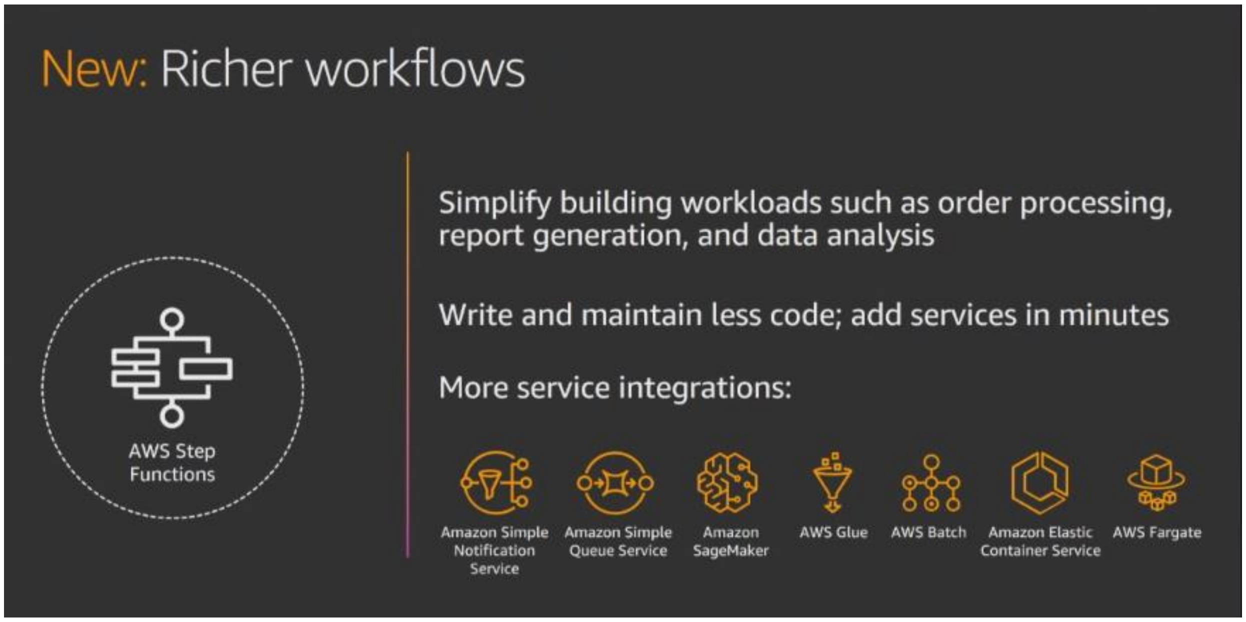
HTTP/2 data retrieval API improves data delivery speed between data producers and Lambda functions by more than 65%



Build workflows to orchestrate everything

Track status of data and execution

Remove redundant code



New: Richer workflows

AWS Step Functions

Simplify building workloads such as order processing, report generation, and data analysis

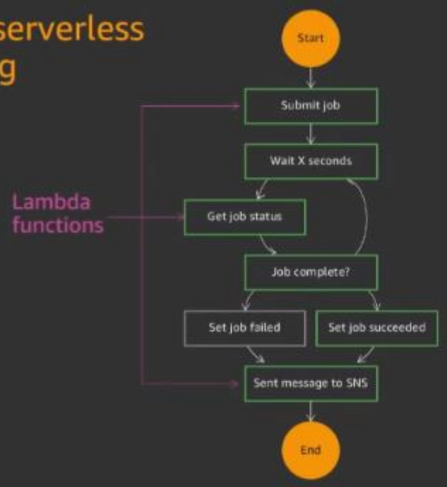
Write and maintain less code; add services in minutes

More service integrations:

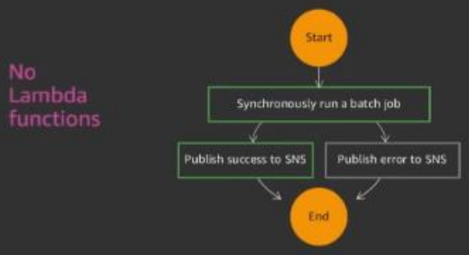
Amazon Simple Notification Service · Amazon Simple Queue Service · Amazon SageMaker · AWS Glue · AWS Batch · Amazon Elastic Container Service · AWS Fargate

Simpler integration, less code

With serverless polling



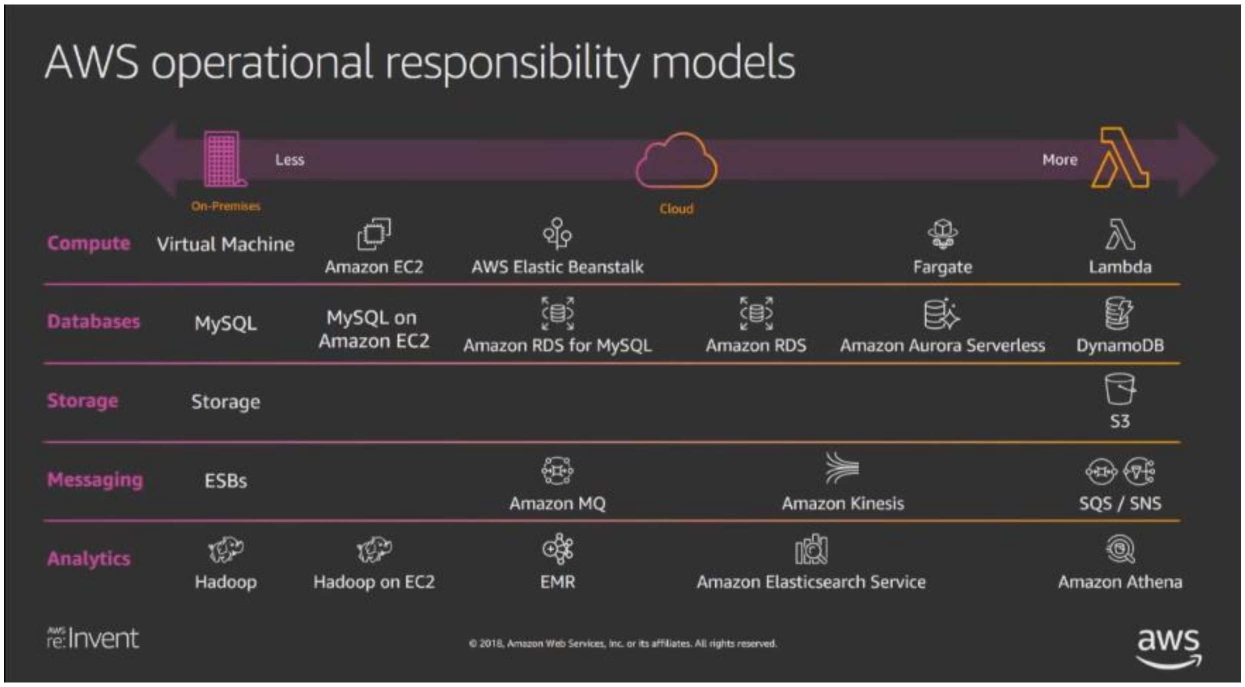
With new service integration



Cloud-native architectures are small pieces, loosely joined

Changes to the operational model

Isn't all of this very hard now that we have lots of pieces to operate?



AWS operational responsibility models

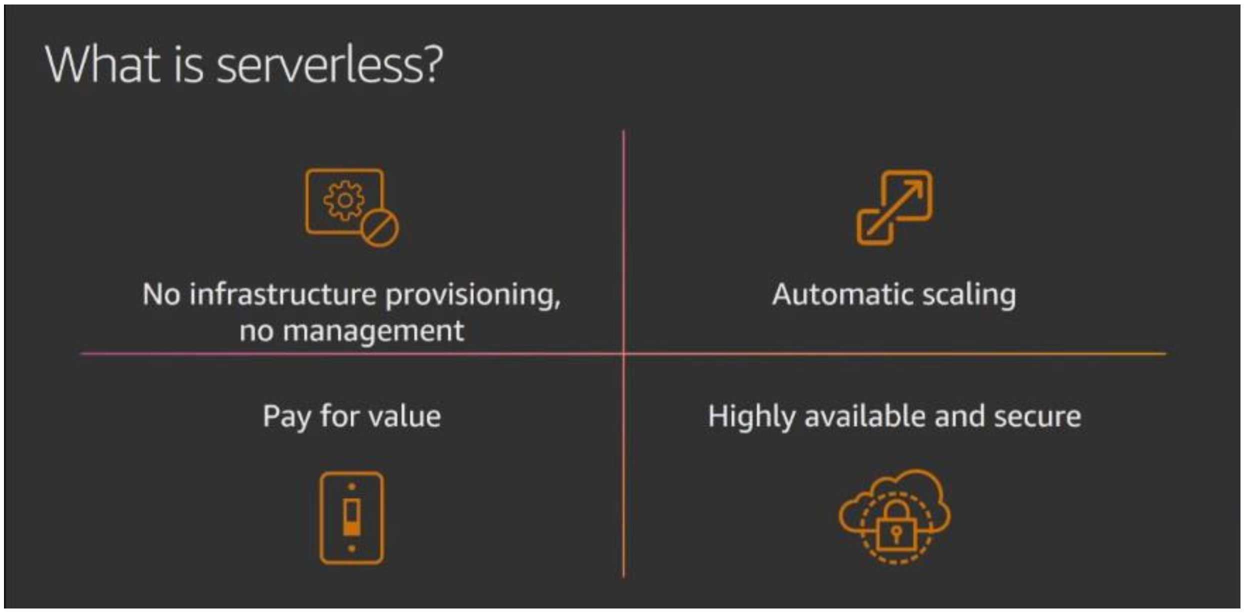
Less
On-Premises
Cloud
More

Compute	Virtual Machine	Amazon EC2	AWS Elastic Beanstalk		Fargate	Lambda
Databases	MySQL	MySQL on Amazon EC2	Amazon RDS for MySQL	Amazon RDS	Amazon Aurora Serverless	DynamoDB
Storage	Storage					S3
Messaging	ESBs		Amazon MQ		Amazon Kinesis	SQS / SNS
Analytics	Hadoop	Hadoop on EC2	EMR	Amazon Elasticsearch Service		Amazon Athena

re:Invent

aws

Cluster huggers are the new server huggers



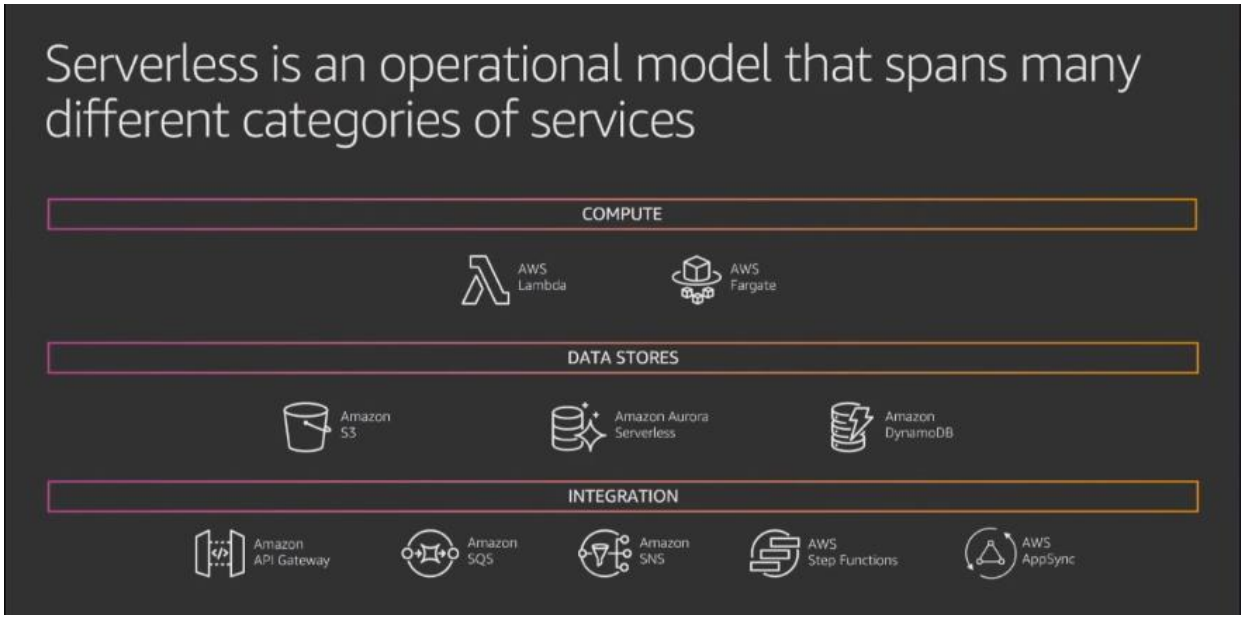
What is serverless?

No infrastructure provisioning, no management

Automatic scaling

Pay for value

Highly available and secure



Serverless is an operational model that spans many different categories of services

COMPUTE

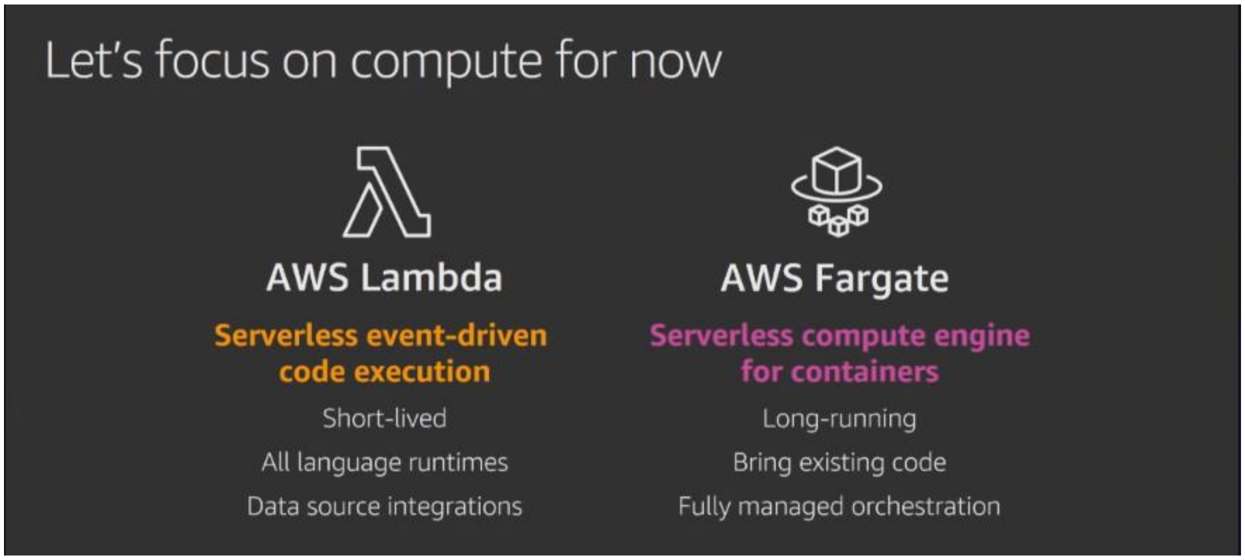
AWS Lambda · AWS Fargate

DATA STORES

Amazon S3 · Amazon Aurora Serverless · Amazon DynamoDB

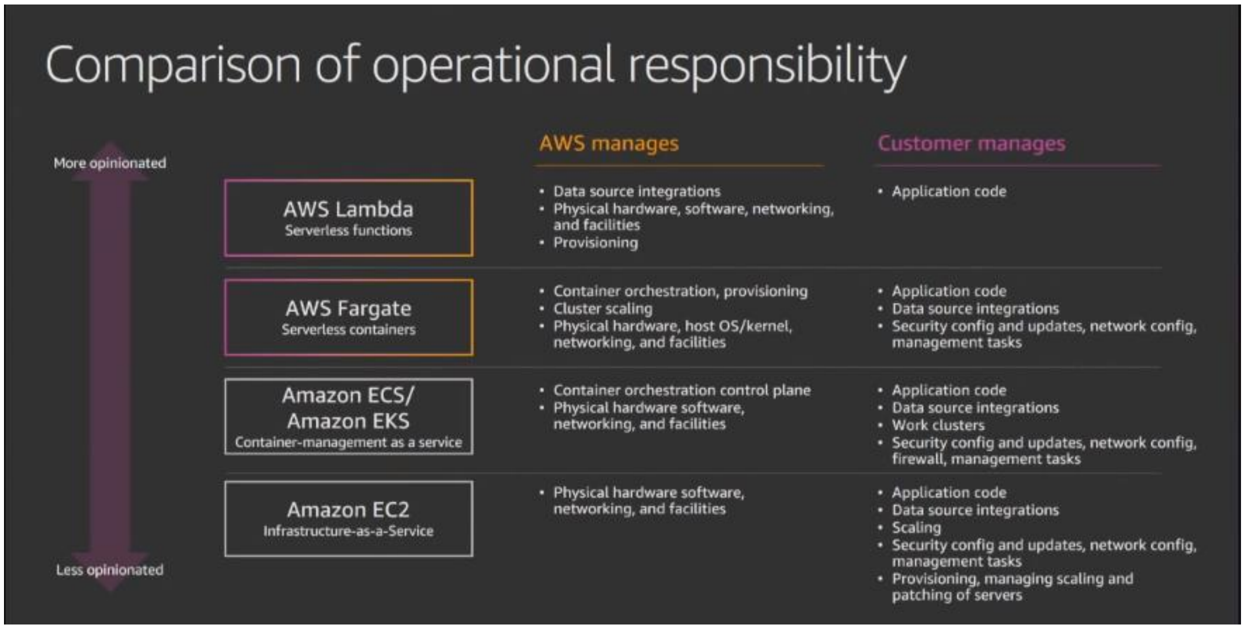
INTEGRATION

Amazon API Gateway · Amazon SQS · Amazon SNS · AWS Step Functions · AWS AppSync



Let's focus on compute for now

AWS Lambda	AWS Fargate
Serverless event-driven code execution	**Serverless compute engine for containers**
Short-lived	Long-running
All language runtimes	Bring existing code
Data source integrations	Fully managed orchestration

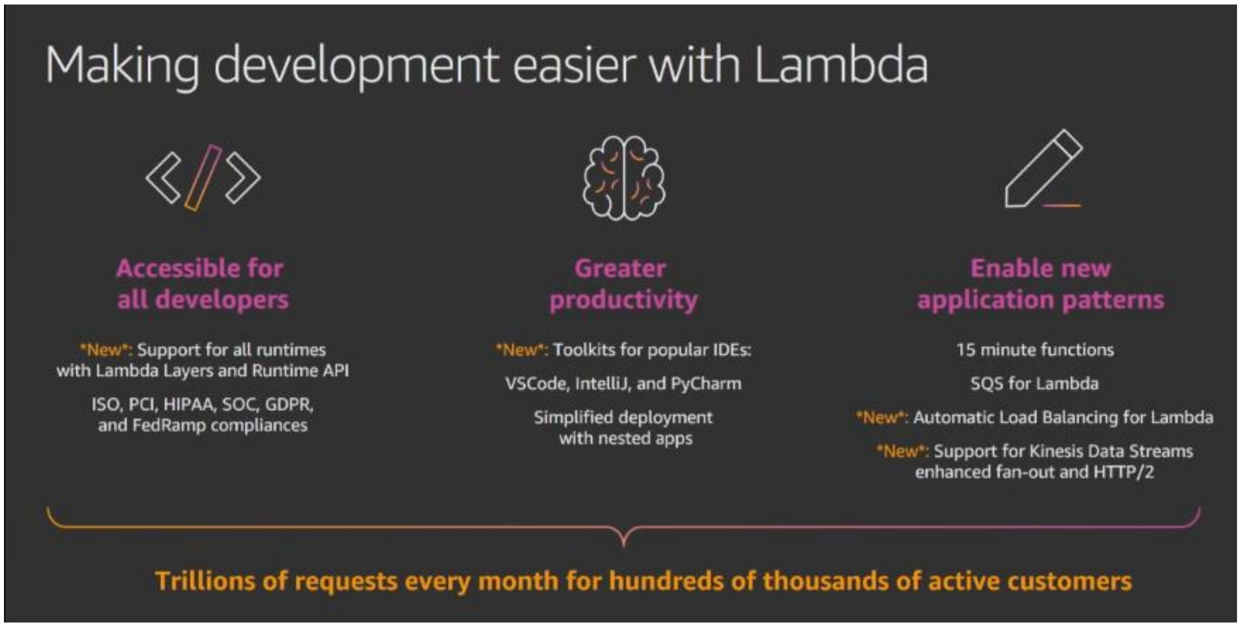


Comparison of operational responsibility

More opinionated

	AWS manages	Customer manages
AWS Lambda Serverless functions	• Data source integrations • Physical hardware, software, networking, and facilities • Provisioning	• Application code
AWS Fargate Serverless containers	• Container orchestration, provisioning • Cluster scaling • Physical hardware, host OS/kernel, networking, and facilities	• Application code • Data source integrations • Security config and updates, network config, management tasks
Amazon ECS/ Amazon EKS Container-management as a service	• Container orchestration control plane • Physical hardware software, networking, and facilities	• Application code • Data source integrations • Work clusters • Security config and updates, network config, firewall, management tasks
Amazon EC2 Infrastructure-as-a-Service	• Physical hardware software, networking, and facilities	• Application code • Data source integrations • Scaling • Security config and updates, network config, management tasks • Provisioning, managing scaling and patching of servers

Less opinionated



Making development easier with Lambda

Accessible for all developers

New: Support for all runtimes with Lambda Layers and Runtime API

ISO, PCI, HIPAA, SOC, GDPR, and FedRamp compliances

Greater productivity

New: Toolkits for popular IDEs:

VSCode, IntelliJ, and PyCharm

Simplified deployment with nested apps

Enable new application patterns

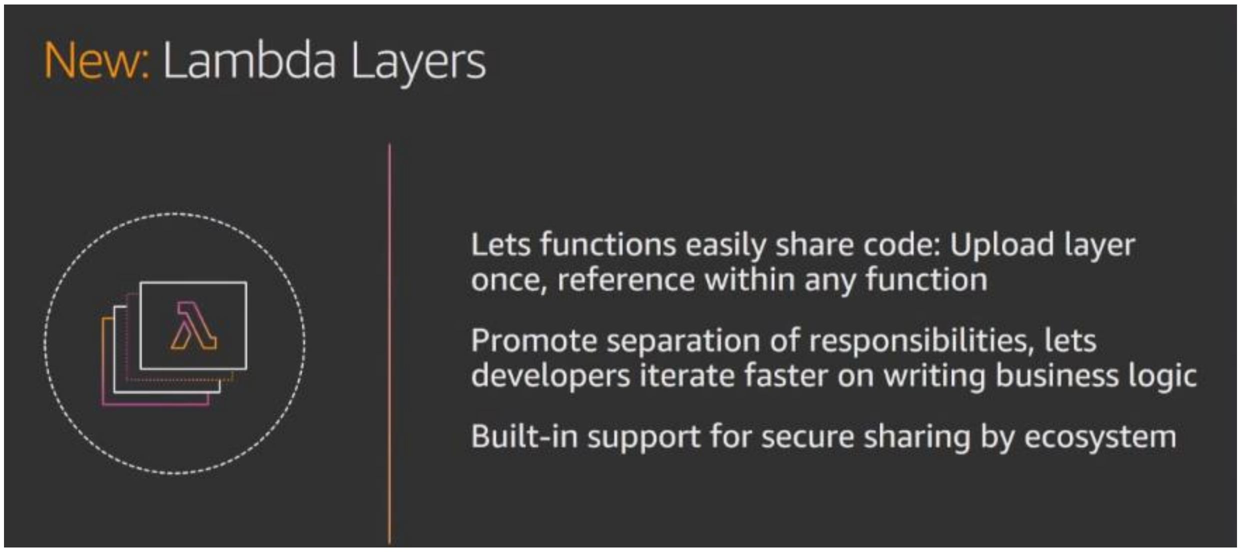
15 minute functions

SQS for Lambda

New: Automatic Load Balancing for Lambda

New: Support for Kinesis Data Streams enhanced fan-out and HTTP/2

Trillions of requests every month for hundreds of thousands of active customers

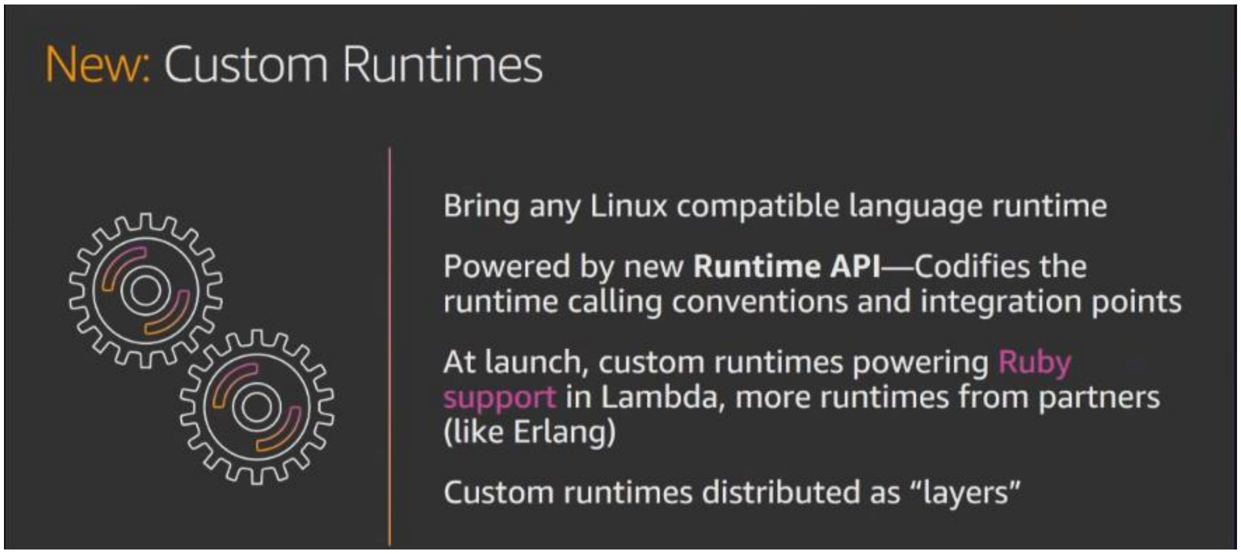


New: Lambda Layers

Lets functions easily share code: Upload layer once, reference within any function

Promote separation of responsibilities, lets developers iterate faster on writing business logic

Built-in support for secure sharing by ecosystem



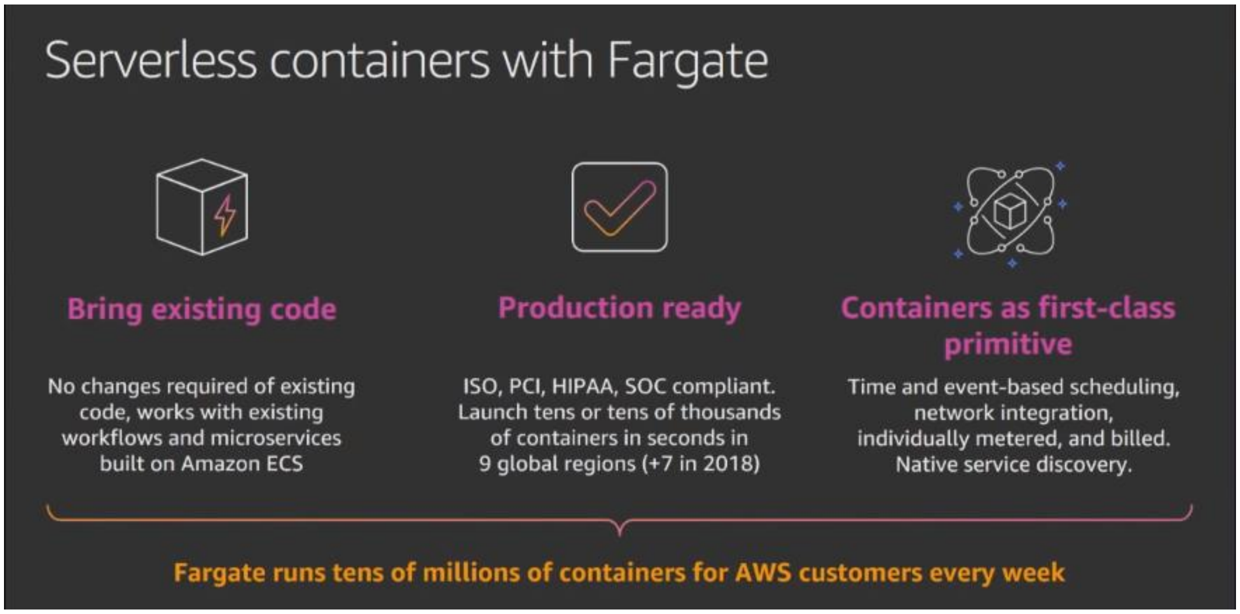
New: Custom Runtimes

Bring any Linux compatible language runtime

Powered by new **Runtime API**—Codifies the runtime calling conventions and integration points

At launch, custom runtimes powering Ruby support in Lambda, more runtimes from partners (like Erlang)

Custom runtimes distributed as “layers”



Serverless containers with Fargate

Bring existing code

No changes required of existing code, works with existing workflows and microservices built on Amazon ECS

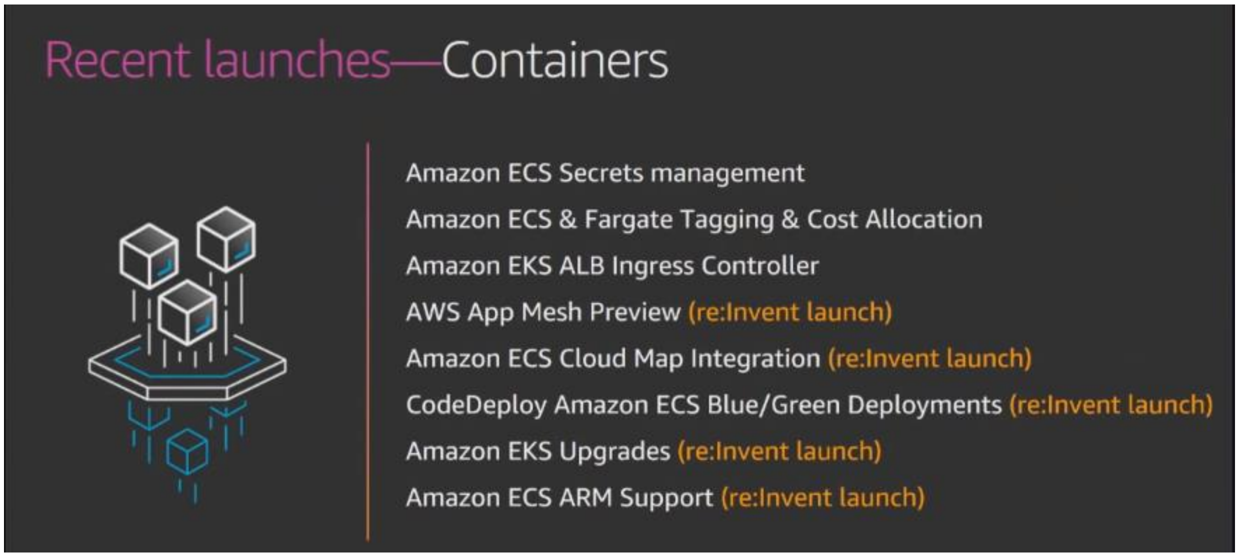
Production ready

ISO, PCI, HIPAA, SOC compliant. Launch tens or tens of thousands of containers in seconds in 9 global regions (+7 in 2018)

Containers as first-class primitive

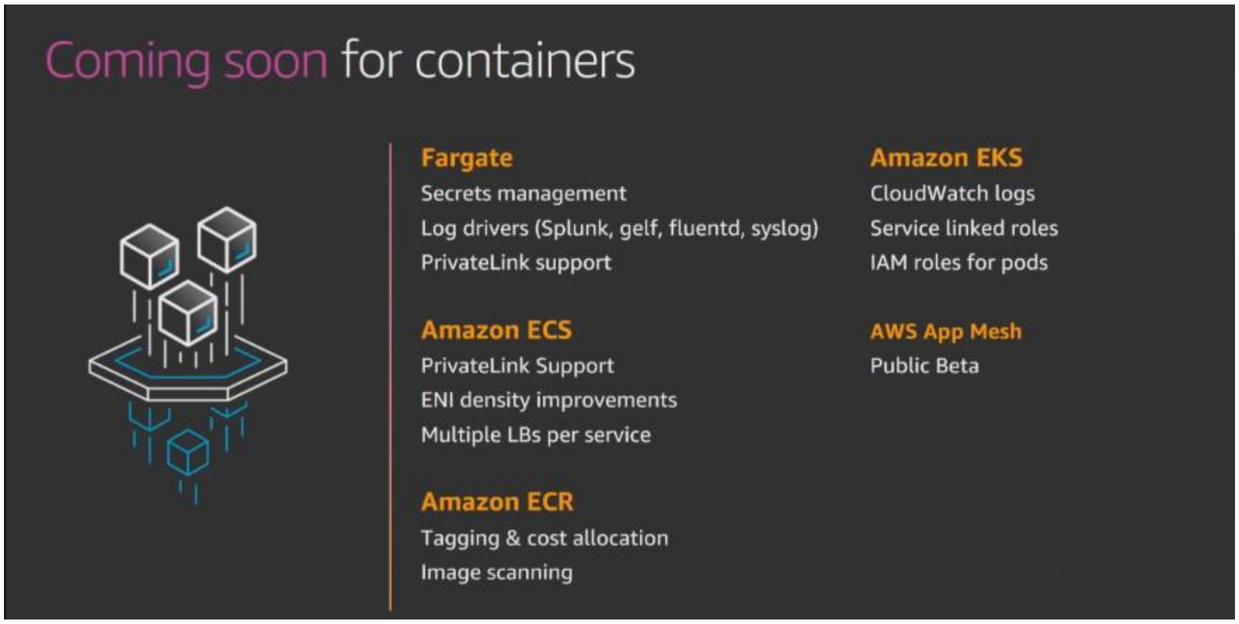
Time and event-based scheduling, network integration, individually metered, and billed. Native service discovery.

Fargate runs tens of millions of containers for AWS customers every week



Recent launches—Containers

- Amazon ECS Secrets management
- Amazon ECS & Fargate Tagging & Cost Allocation
- Amazon EKS ALB Ingress Controller
- AWS App Mesh Preview (re:Invent launch)
- Amazon ECS Cloud Map Integration (re:Invent launch)
- CodeDeploy Amazon ECS Blue/Green Deployments (re:Invent launch)
- Amazon EKS Upgrades (re:Invent launch)
- Amazon ECS ARM Support (re:Invent launch)



Coming soon for containers

Fargate

Secrets management
Log drivers (Splunk, gelf, fluentd, syslog)
PrivateLink support

Amazon ECS

PrivateLink Support
ENI density improvements
Multiple LBs per service

Amazon ECR

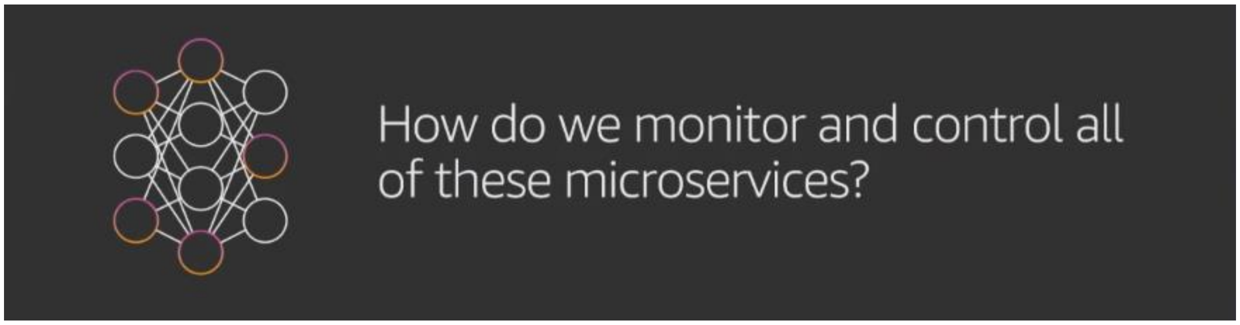
Tagging & cost allocation
Image scanning

Amazon EKS

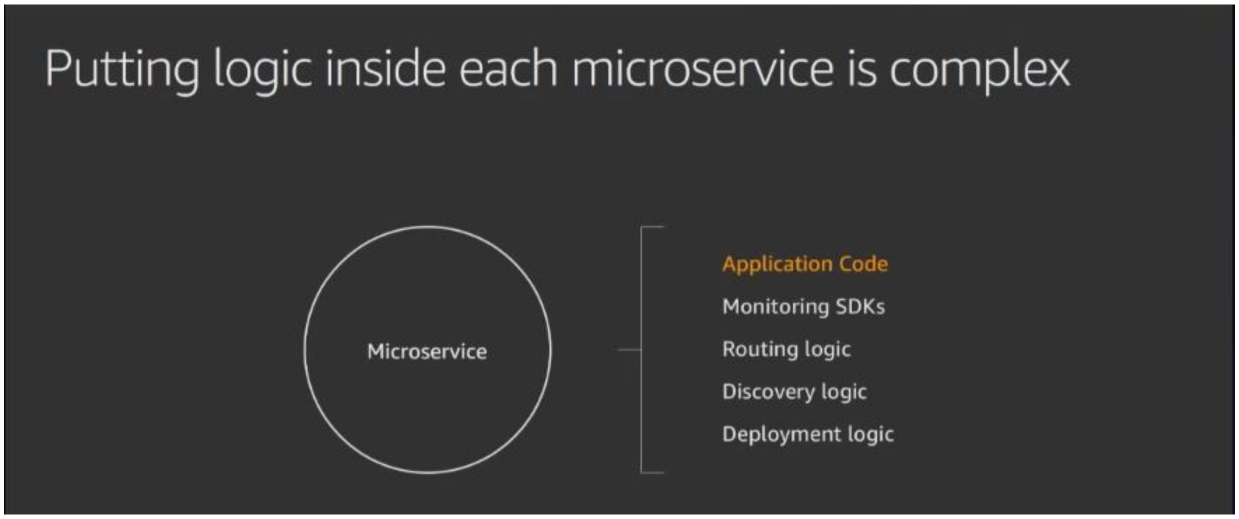
CloudWatch logs
Service linked roles
IAM roles for pods

AWS App Mesh

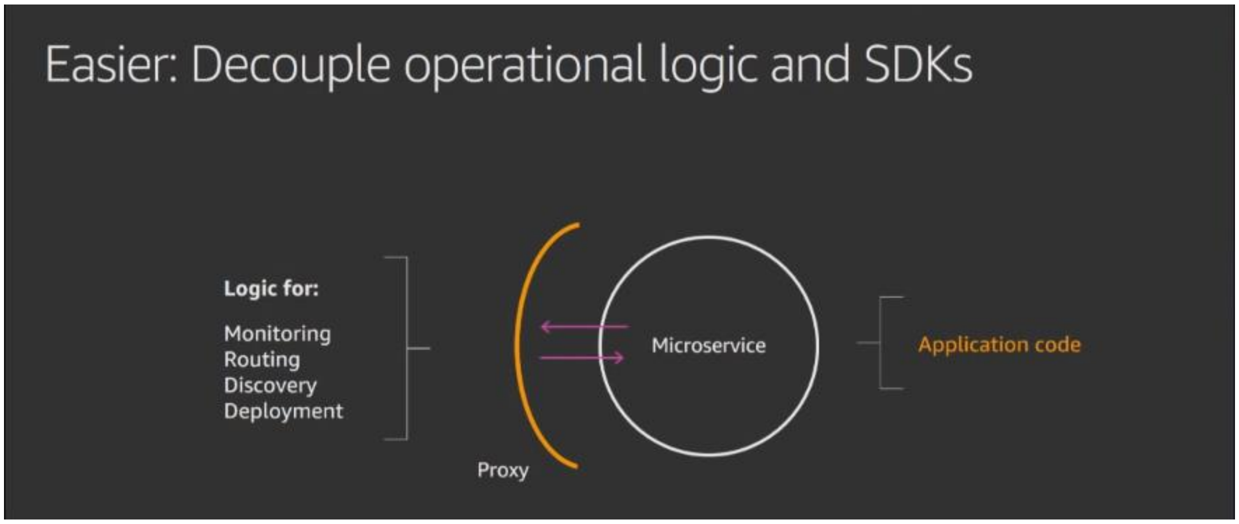
Public Beta



How do we monitor and control all of these microservices?

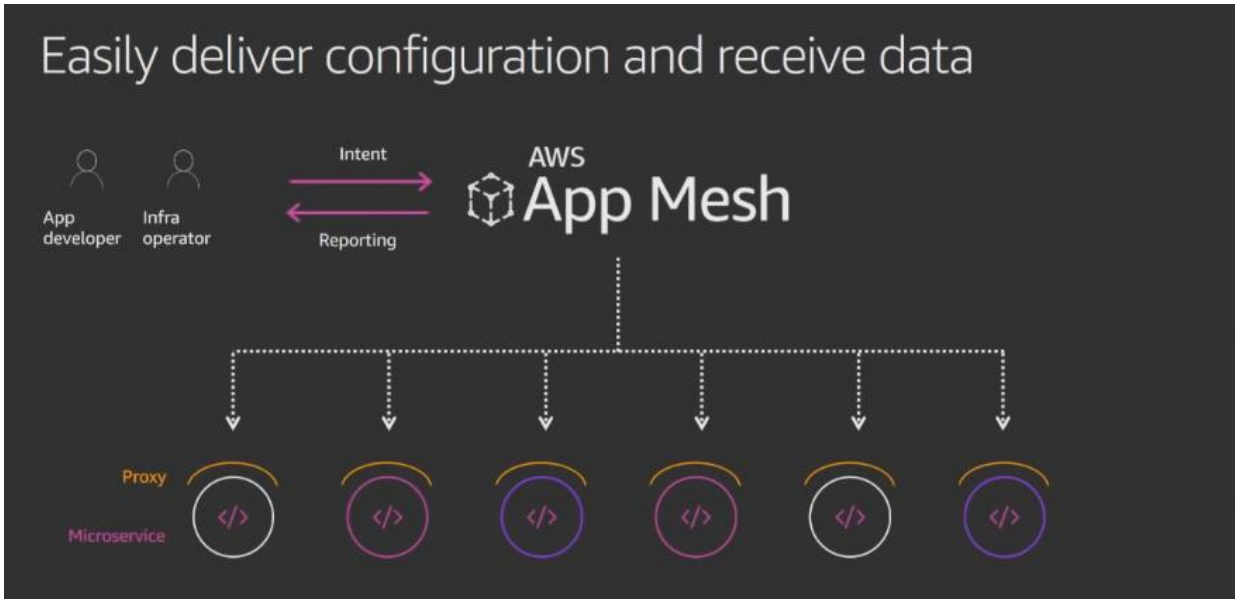


Putting logic inside each microservice is complex



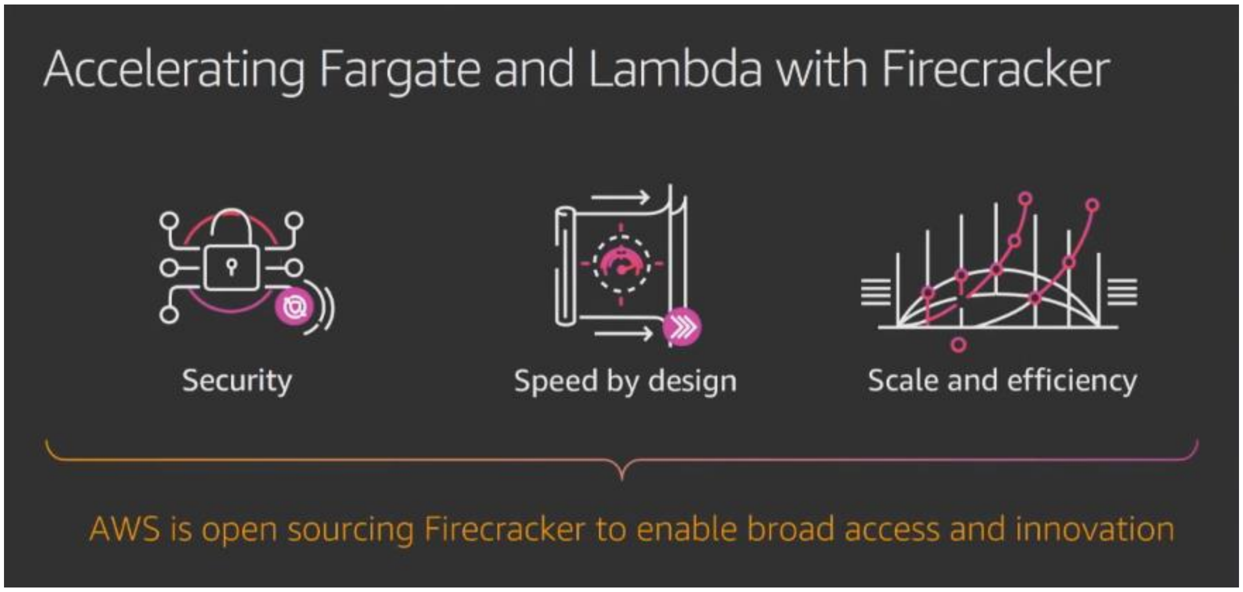
Easier: Decouple operational logic and SDKs

You deploy a side car proxy that your application code then talks to when interacting with the microservice



Easily deliver configuration and receive data

AWS App Mesh is a data plane for Envoy side car proxy for your microservices, it acts as a centralized control plane for all your microservices running within your architecture.

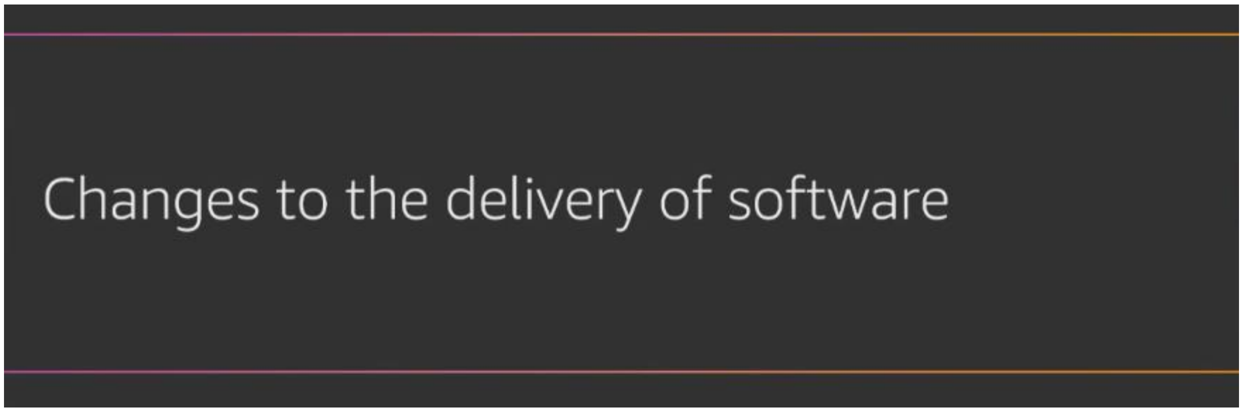


Accelerating Fargate and Lambda with Firecracker

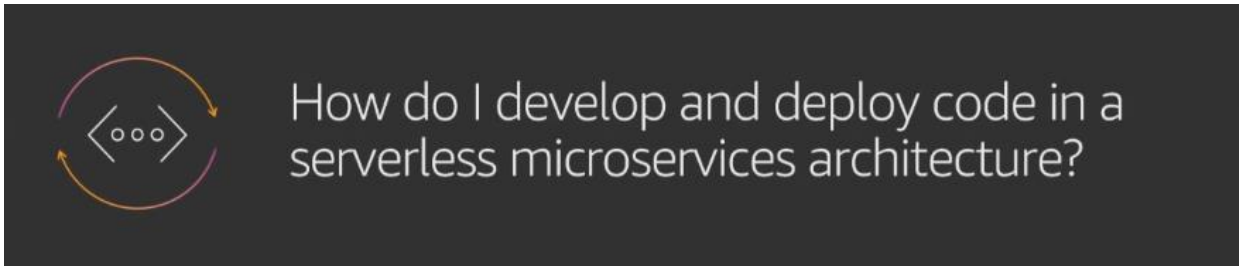
Security
Speed by design
Scale and efficiency

AWS is open sourcing Firecracker to enable broad access and innovation

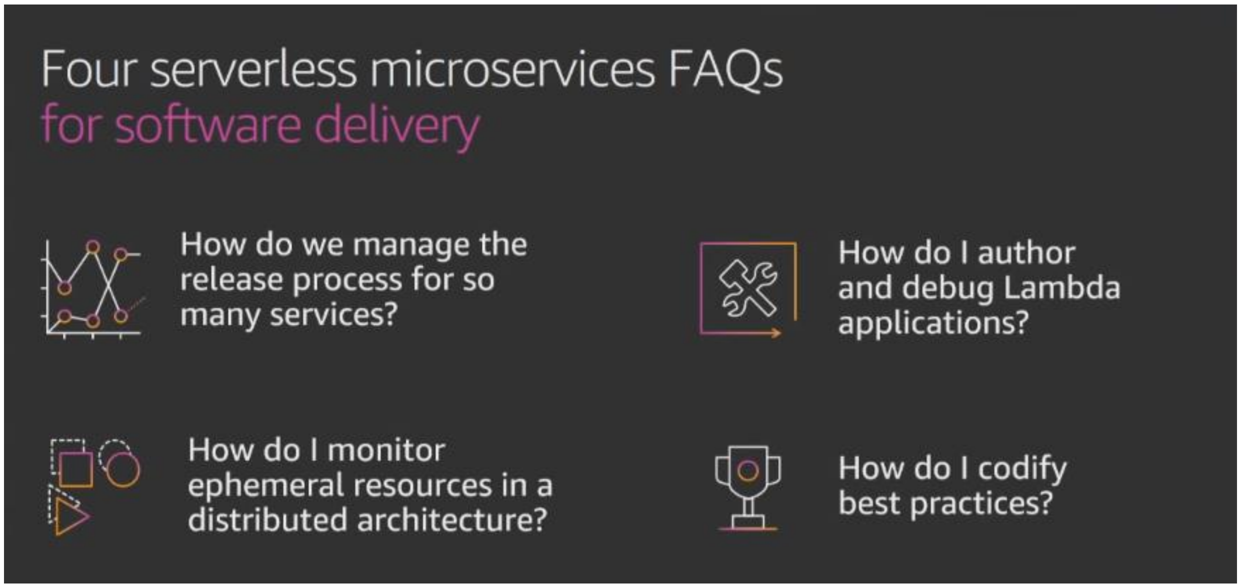
Firecracker is a little VM monitor that allows you to run lots of containers on your EC2 instances.



Changes to the delivery of software

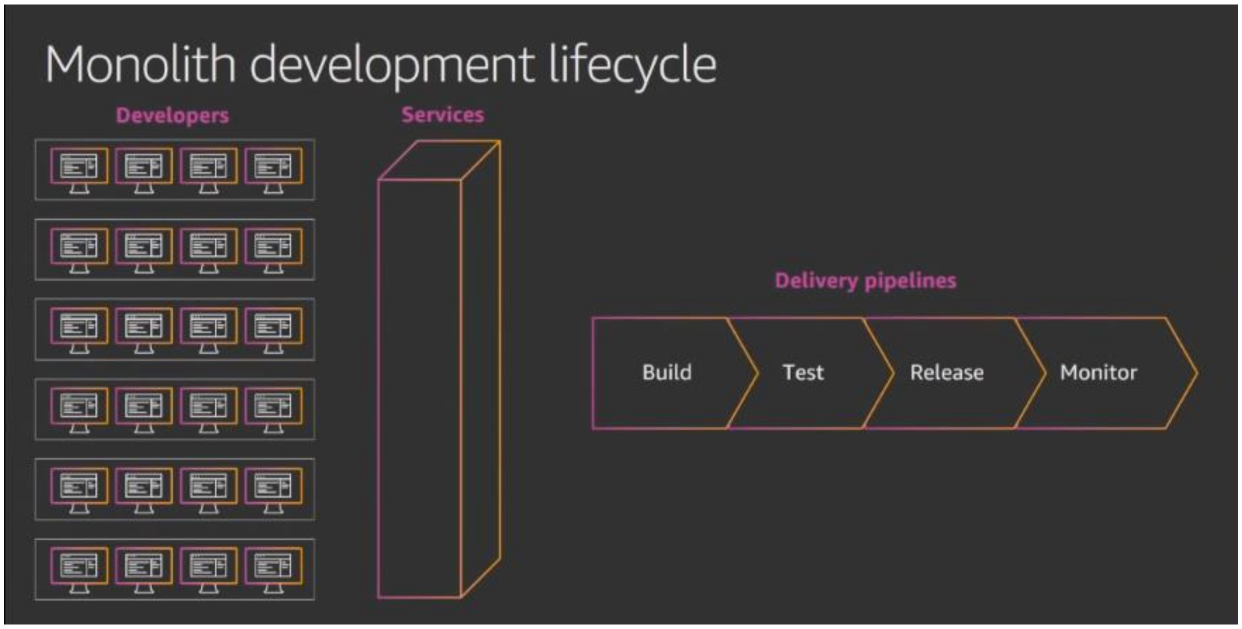


How do I develop and deploy code in a serverless microservices architecture?

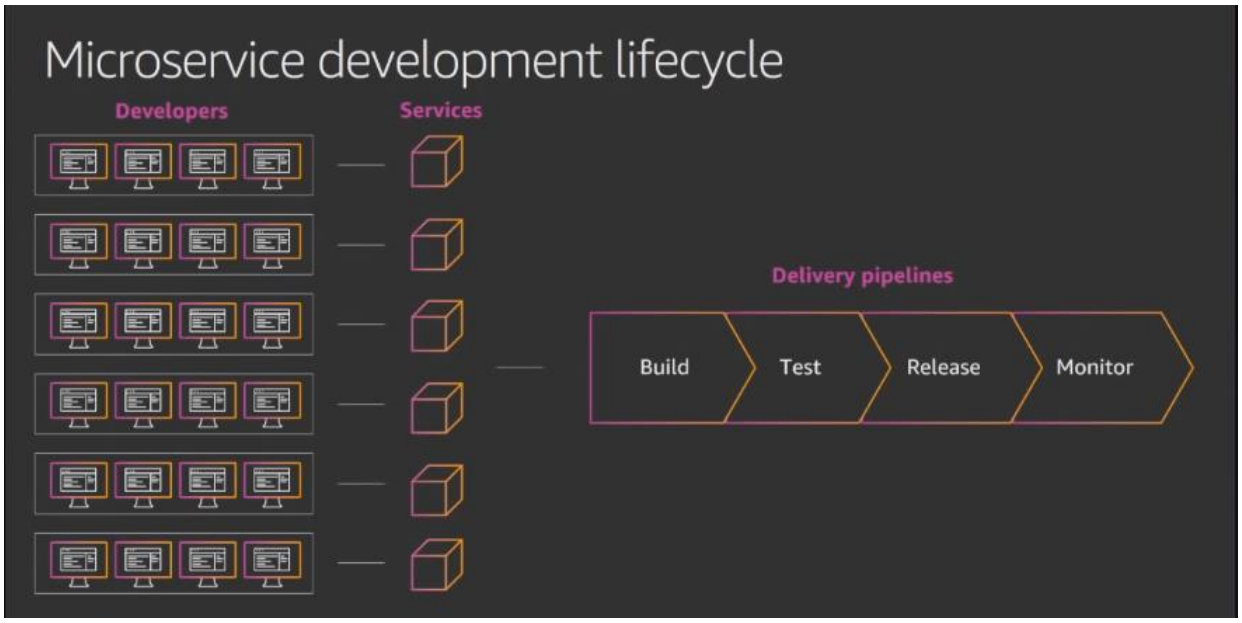


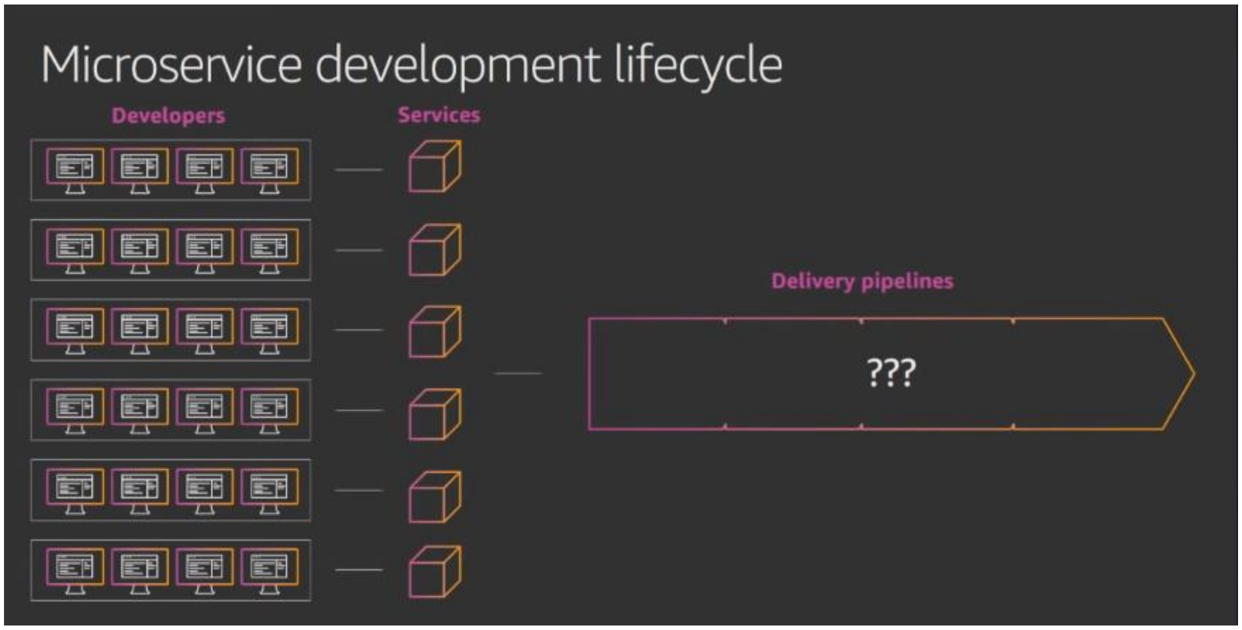
Four serverless microservices FAQs for software delivery

- How do we manage the release process for so many services?
- How do I author and debug Lambda applications?
- How do I monitor ephemeral resources in a distributed architecture?
- How do I codify best practices?

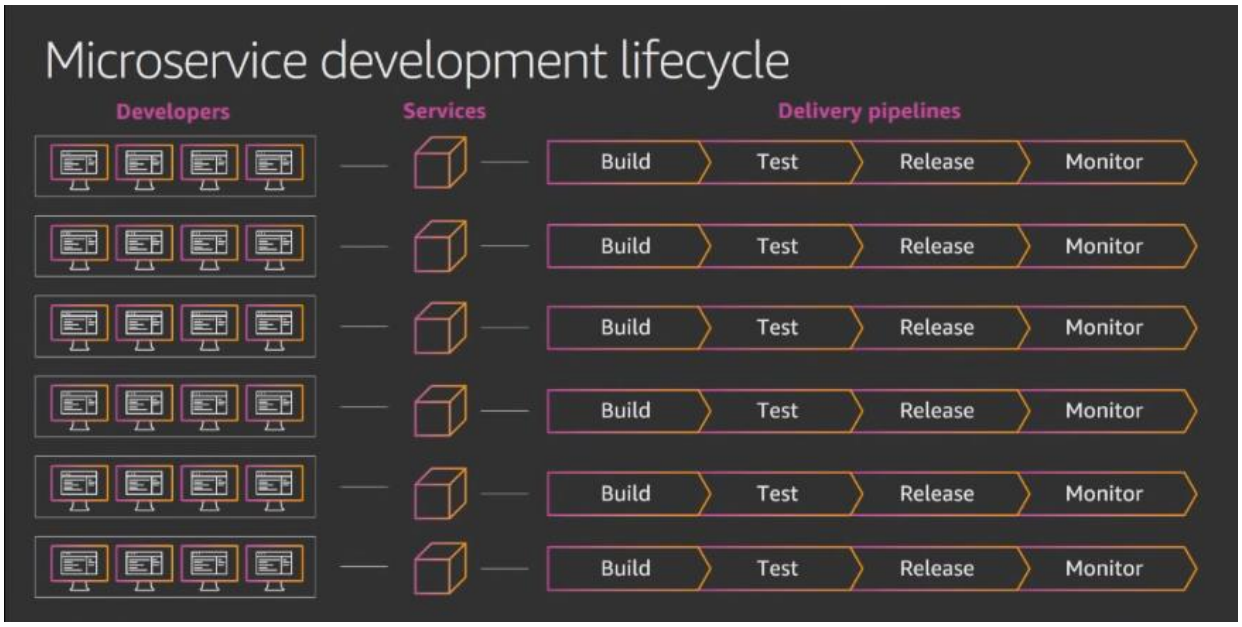


Monolith development lifecycle

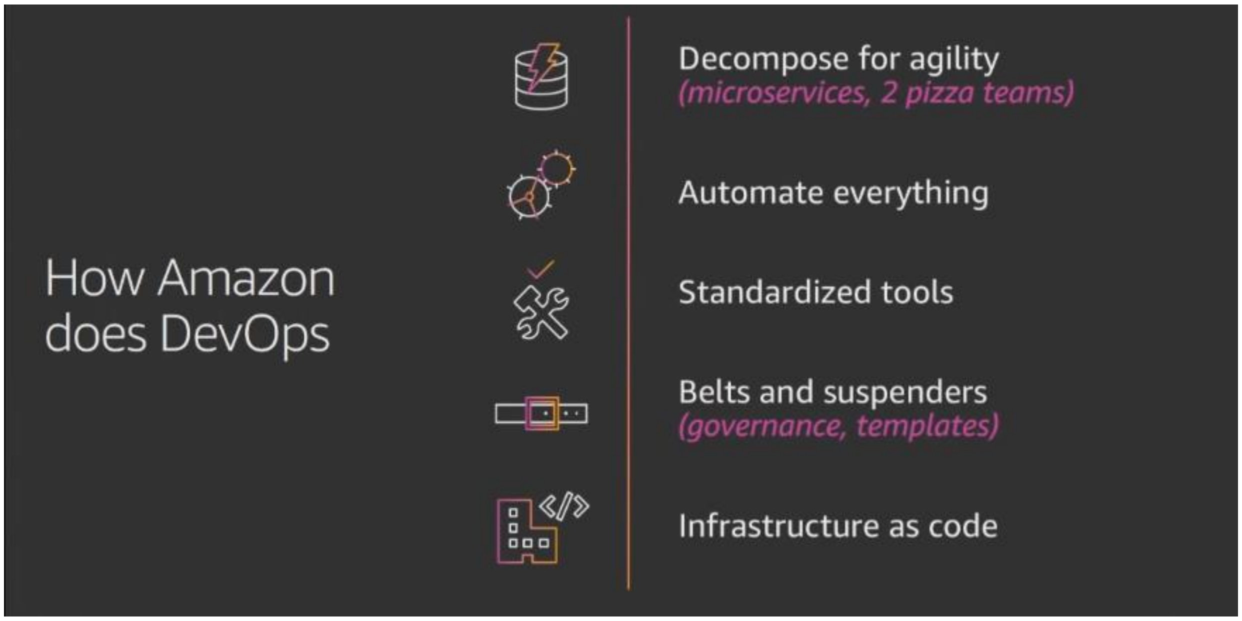




How do you actually release software to production?


Microservice development lifecycle

We gave each DEV team the tools to do their own deployment and release process


How Amazon does DevOps

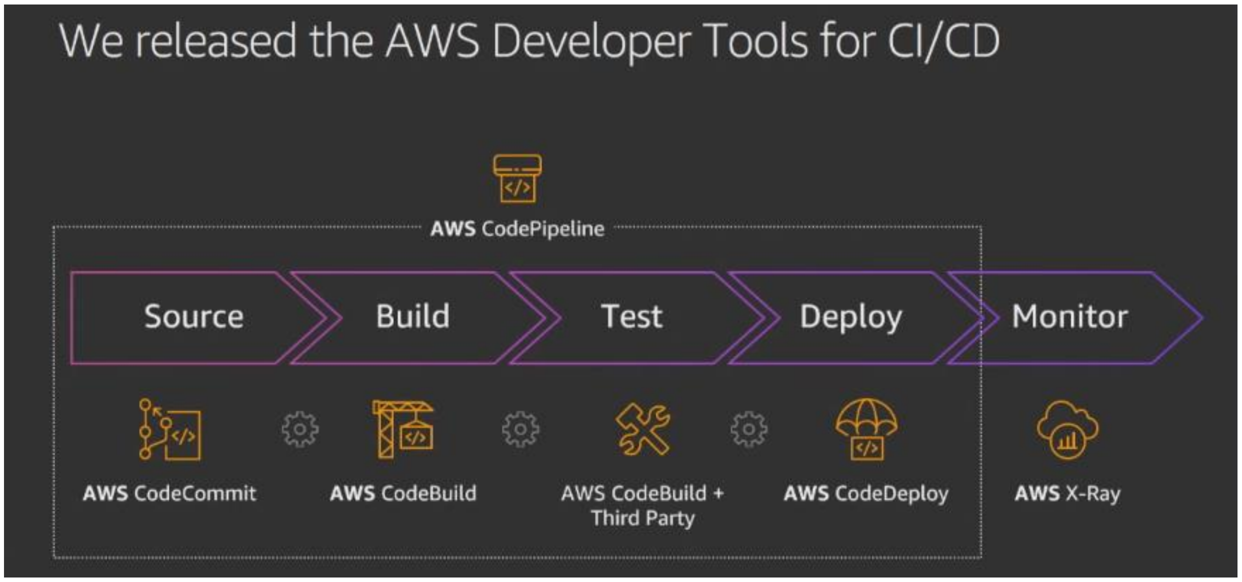
Decompose for agility
(microservices, 2 pizza teams)

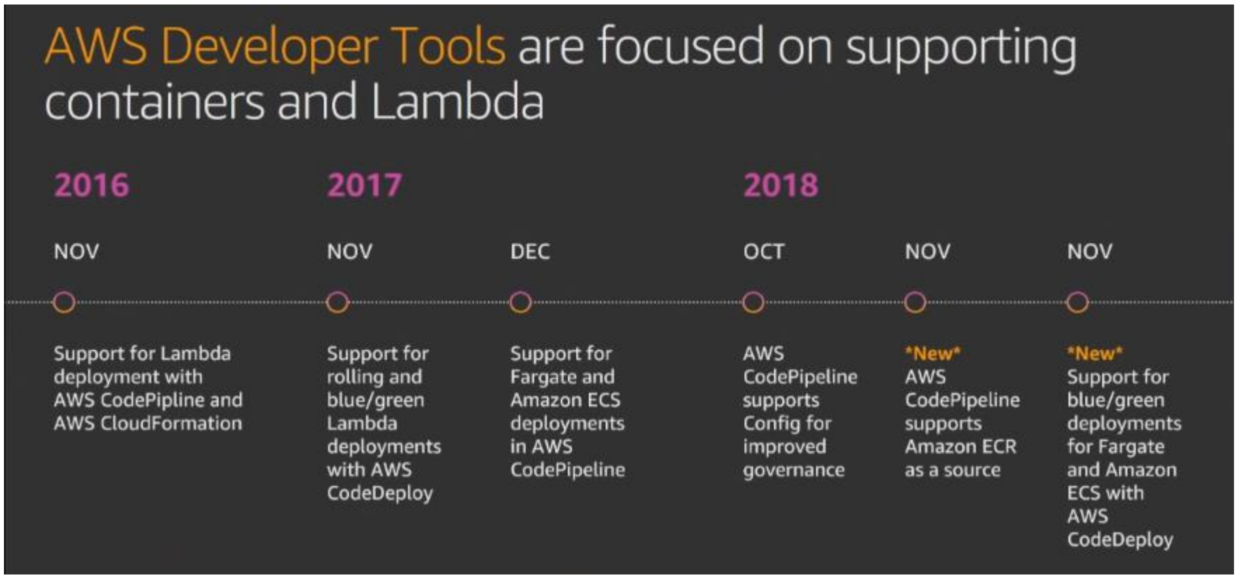
Automate everything

Standardized tools

Belts and suspenders
(governance, templates)

Infrastructure as code

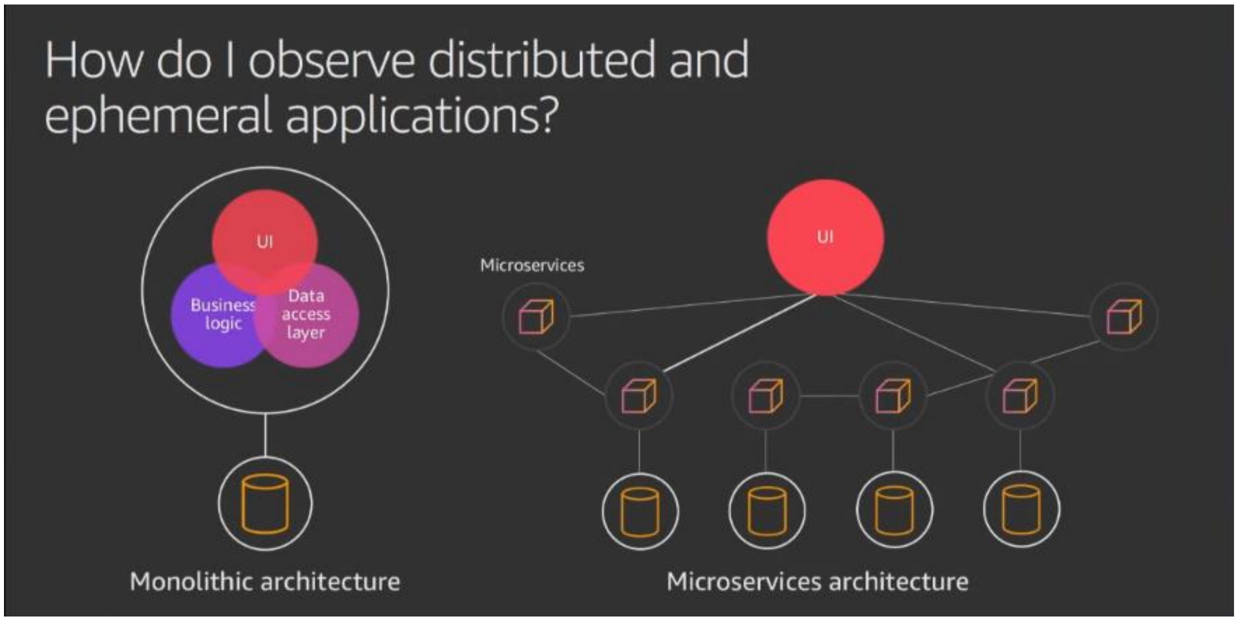

We released the AWS Developer Tools for CI/CD

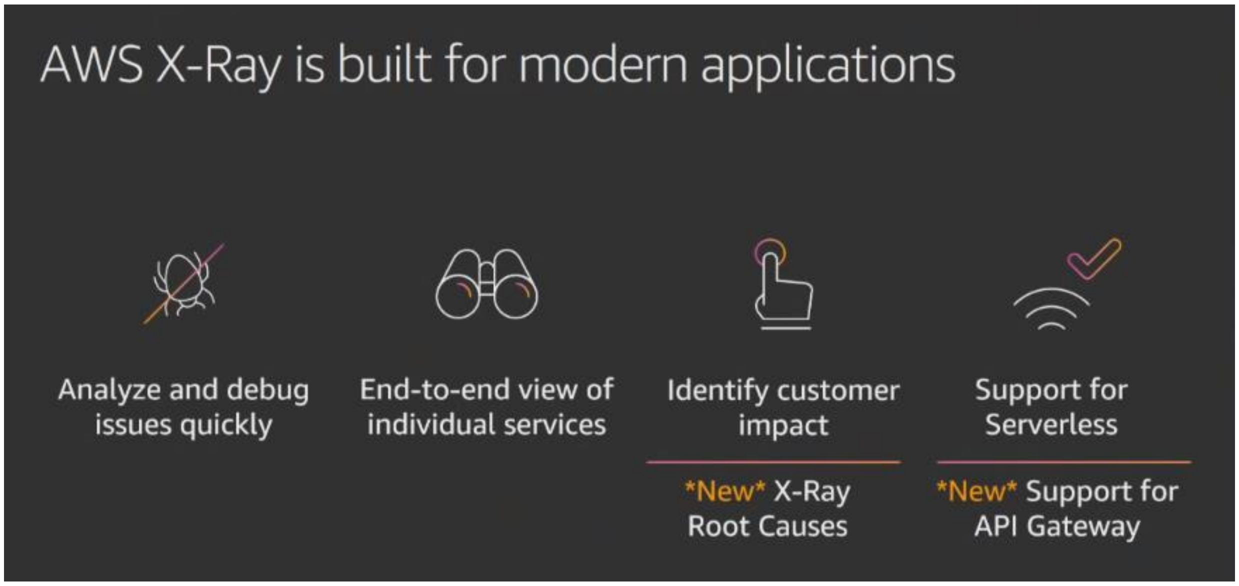


AWS Developer Tools are focused on supporting containers and Lambda

2016	2017		2018		
NOV	NOV	DEC	OCT	NOV	NOV
Support for Lambda deployment with AWS CodePipline and AWS CloudFormation	Support for rolling and blue/green Lambda deployments with AWS CodeDeploy	Support for Fargate and Amazon ECS deployments in AWS CodePipeline	AWS CodePipeline supports Config for improved governance	*New* AWS CodePipeline supports Amazon ECR as a source	*New* Support for blue/green deployments for Fargate and Amazon ECS with AWS CodeDeploy

How do I observe distributed and ephemeral applications?





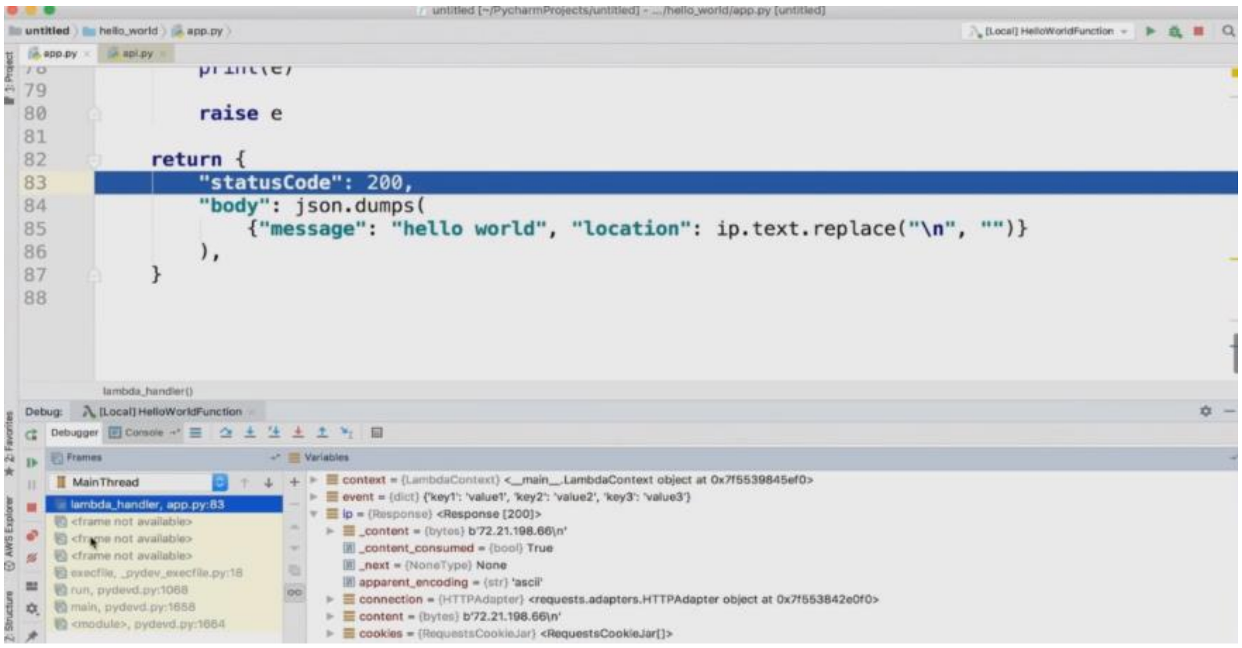
AWS X-Ray is built for modern applications

- Analyze and debug issues quickly
- End-to-end view of individual services
- Identify customer impact
  - *New* X-Ray Root Causes
- Support for Serverless
  - *New* Support for API Gateway

How do I edit and debug my serverless application code?

Author and debug Lambda applications on AWS using your favorite IDEs

	New Today	Developer Preview	Developer Preview
AWS Cloud9	AWS Toolkit for PyCharm	AWS Toolkit for IntelliJ	AWS Toolkit for Visual Studio Code
Python, Node	Python	Java, Python	.NET, Node



untitled [~/PycharmProjects/untitled] - .../hello_world/app.py [untitled]

untitled › hello_world › app.py

[Local] HelloWorldFunction

app.py × api.py ×

```
            print(e)

            raise e

    return {
        "statusCode": 200,
        "body": json.dumps(
            {"message": "hello world", "location": ip.text.replace("\n", "")}
        ),
    }

```

lambda_handler()

Debug: [Local] HelloWorldFunction

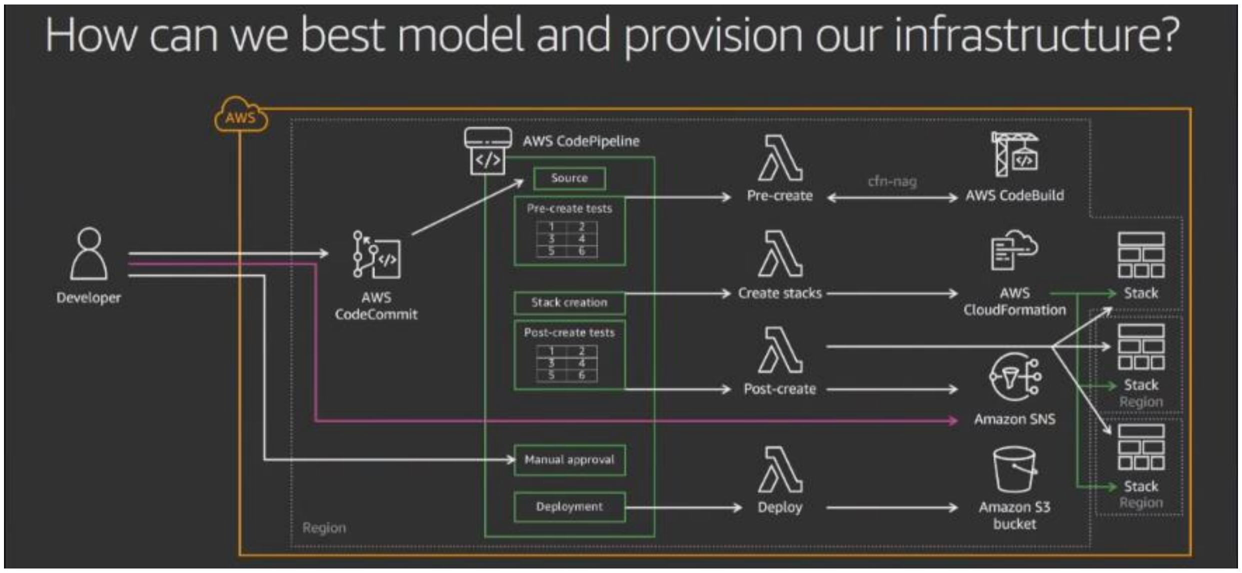
Debugger Console

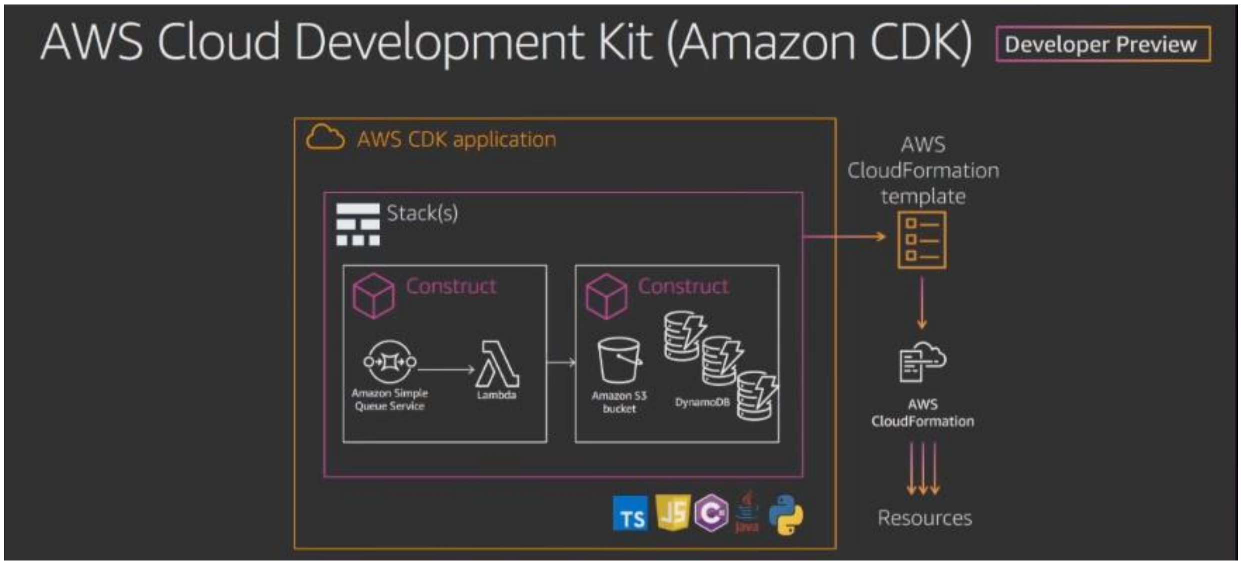
Frames

```
MainThread
lambda_handler, app.py:83
<frame not available>
<frame not available>
<frame not available>
execfile, _pydev_execfile.py:18
run, pydevd.py:1068
main, pydevd.py:1658
<module>, pydevd.py:1664
```

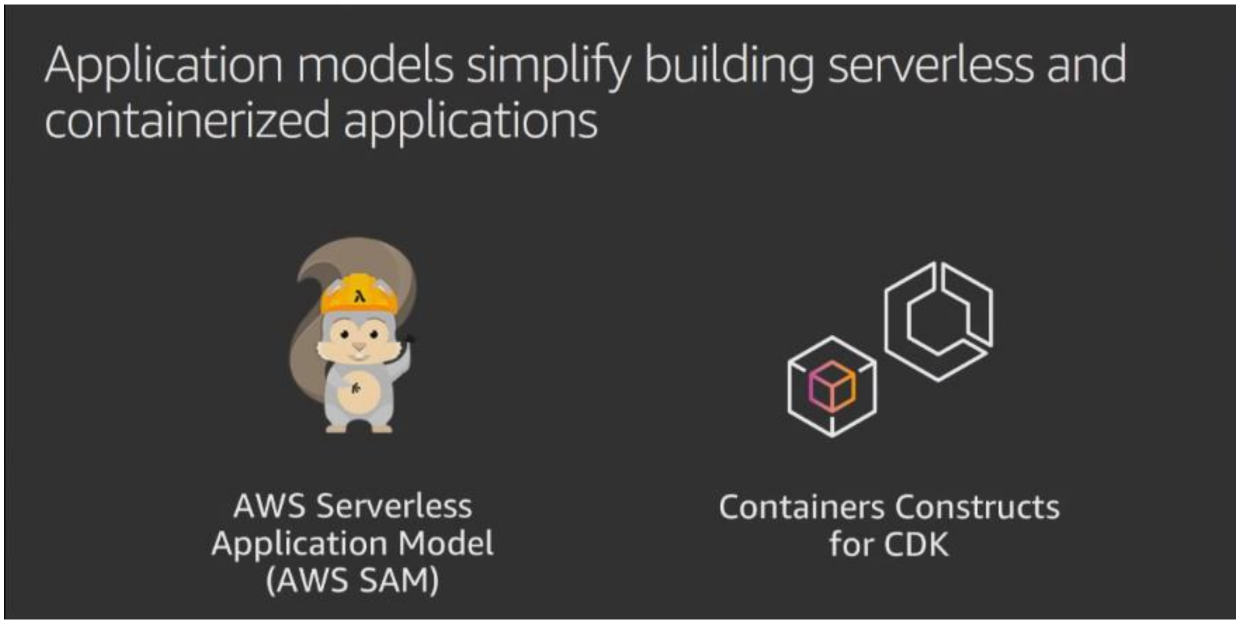
Variables

```
▶ context = {LambdaContext} <__main__.LambdaContext object at 0x7f5539845ef0>
▶ event = {dict} {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
▼ ip = {Response} <Response [200]>
    ▶ _content = {bytes} b'72.21.198.66\n'
      _content_consumed = {bool} True
      _next = {NoneType} None
      apparent_encoding = {str} 'ascii'
    ▶ connection = {HTTPAdapter} <requests.adapters.HTTPAdapter object at 0x7f553842e0f0>
    ▶ content = {bytes} b'72.21.198.66\n'
    ▶ cookies = {RequestsCookieJar} <RequestsCookieJar[]>
```


How can we best model and provision our infrastructure?

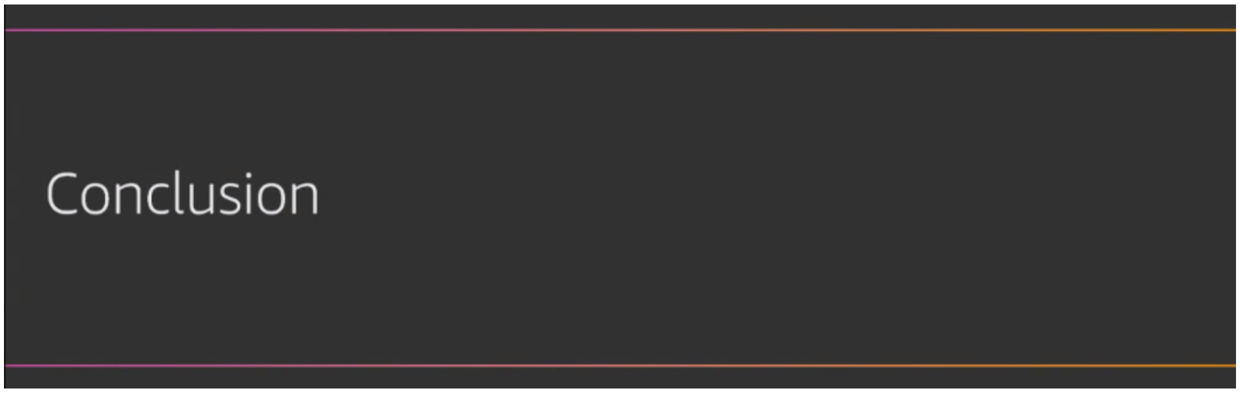

AWS Cloud Development Kit (Amazon CDK) Developer Preview

The CDK provides language bindings for you to author CloudFormation language in imperative languages like Java or TypeScript that compiles down into CF templates.


Application models simplify building serverless and containerized applications

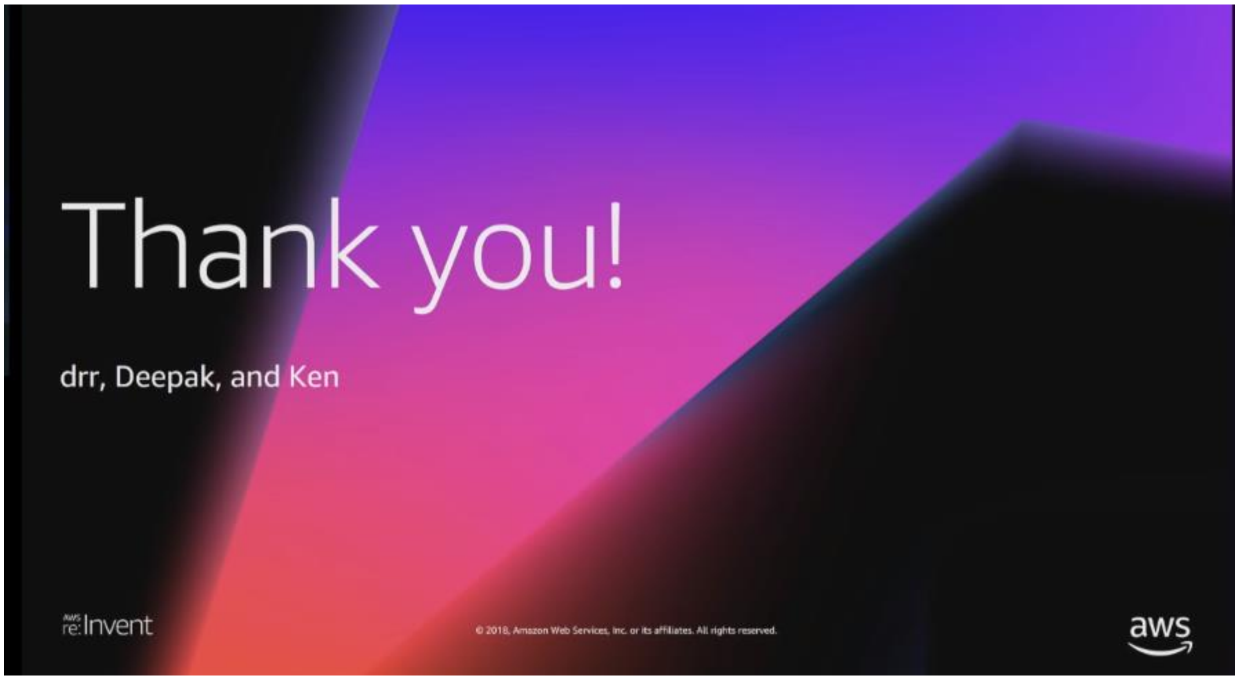
AWS Serverless Application Model (AWS SAM)

Containers Constructs for CDK



Conclusion

We are building a cloud that best supports your modern application development needs, and we are innovating across the entire stack: From the hypervisor layer to the application construction layer.



Thank you!

drr, Deepak, and Ken

re:Invent

aws