AWS

re:Invent

SVS219-S

Serverless at scale

Will Hattingh
Distinguished Engineer, Architect
Capital One

Tanusree McCabe
Distinguished Engineer, Architect
Capital One

The picture can't be displayed.

Agenda

What is serverless?

Patterns

Conclusion

What is serverless?

The picture can't be displayed.

A service is serverless if the following apply

- “No” servers
  - There are no servers exposed that need to be **directly** administered
- Elastic
  - Service scales automatically and is highly available
- Pay as you go
  - You only pay for what you use

“Managed” services are similar but still require the user to perform some server administration (e.g., Amazon ECS)

Representative serverless offerings

Compute	Integration	Analytics	Monitoring	Database	Storage	Mobile
AWS Fargate	AWS Step Functions	Amazon Kinesis	Amazon CloudWatch	Amazon DynamoDB	Amazon S3	Amazon API Gateway
AWS Batch	Amazon SNS	Amazon Athena	AWS X-Ray	AWS Aurora Serverless		
AWS Lambda	Amazon SQS	AWS Glue				

Serverless is applicable for web-based applications, real-time analytics, and processing

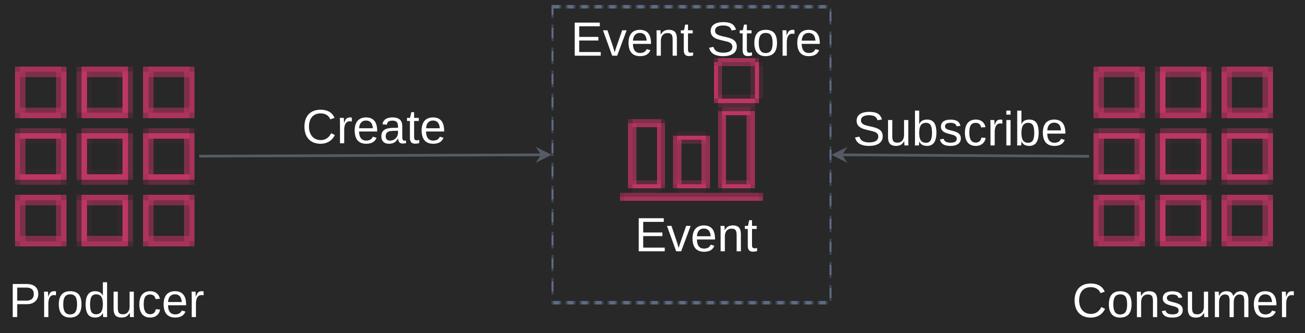
Capital One by the numbers

- Thousands to hundreds of thousands of AWS Lambda functions
  - Multi-regional footprint
- Many PBs of data
  - Many TB/day ingestion
- Thousands of serverless applications
  - Numerous environments

Patterns

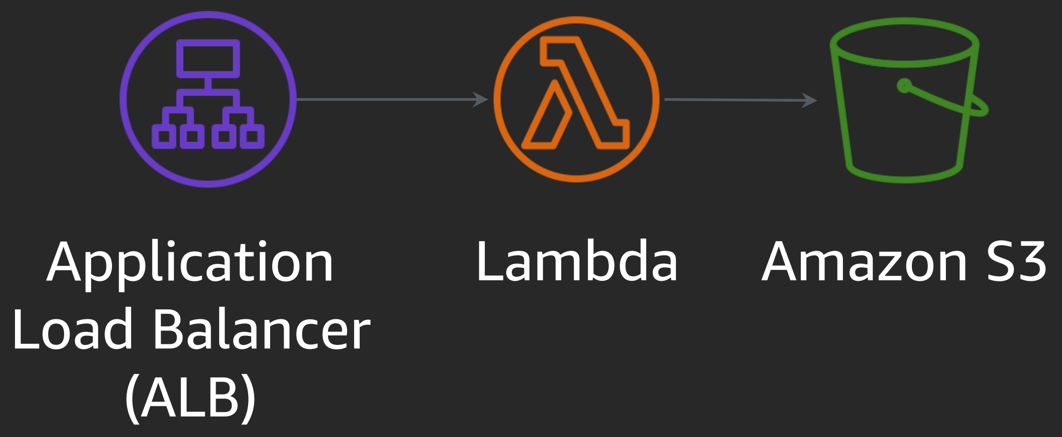
The picture can't be displayed.

Event-based architecture suits serverless



- Asynchronous call enables decoupled systems
- Enables immutable, persistent, shareable events
- Highly resilient to failure
- Able to scale effectively
- Highly observable and extensible system
- Independently releasable
- Independently optimizable

Event-based IRL: Static website hosting



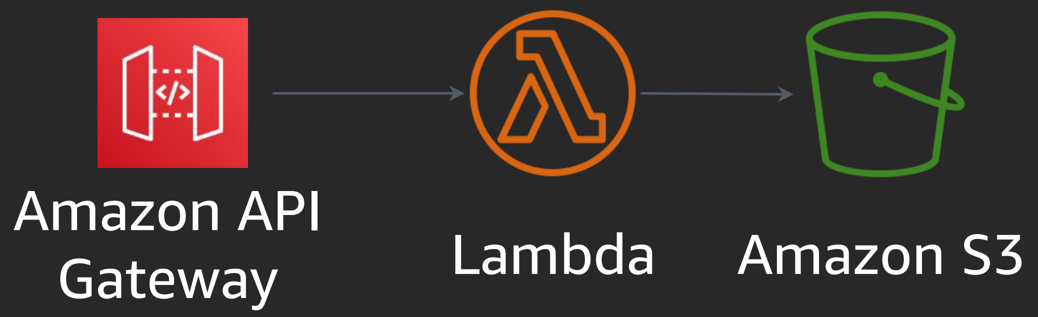
- Private static websites
- Private single-page application (SPA)

Lessons learned

- ALBs have limits
- Streaming is not available
- Gzip is your friend

- Hot Lambda shares memory
- Caching reduces calls

Event-based IRL: API Server



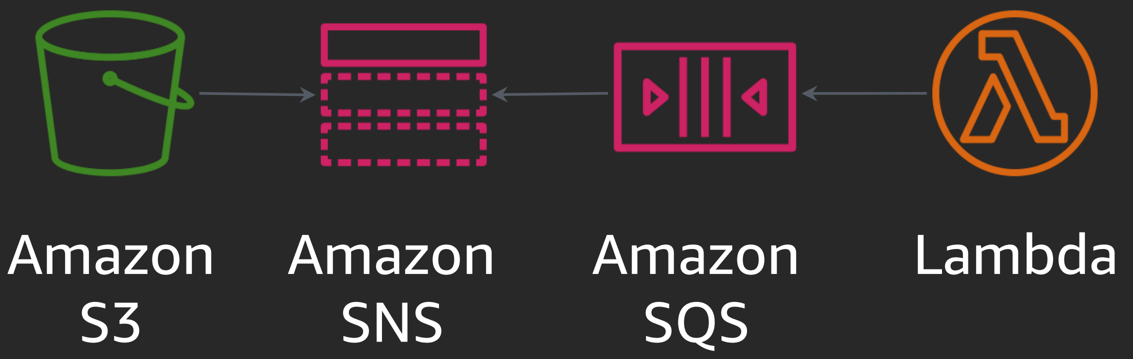
Data processing

Lessons learned

- ALBs have limits
- Gzip is your friend

- Co-location of API server and ETL job reduces latency and improves security
- Amazon API Gateway private link policy improves security

Event-based IRL: Event-driven Lambda

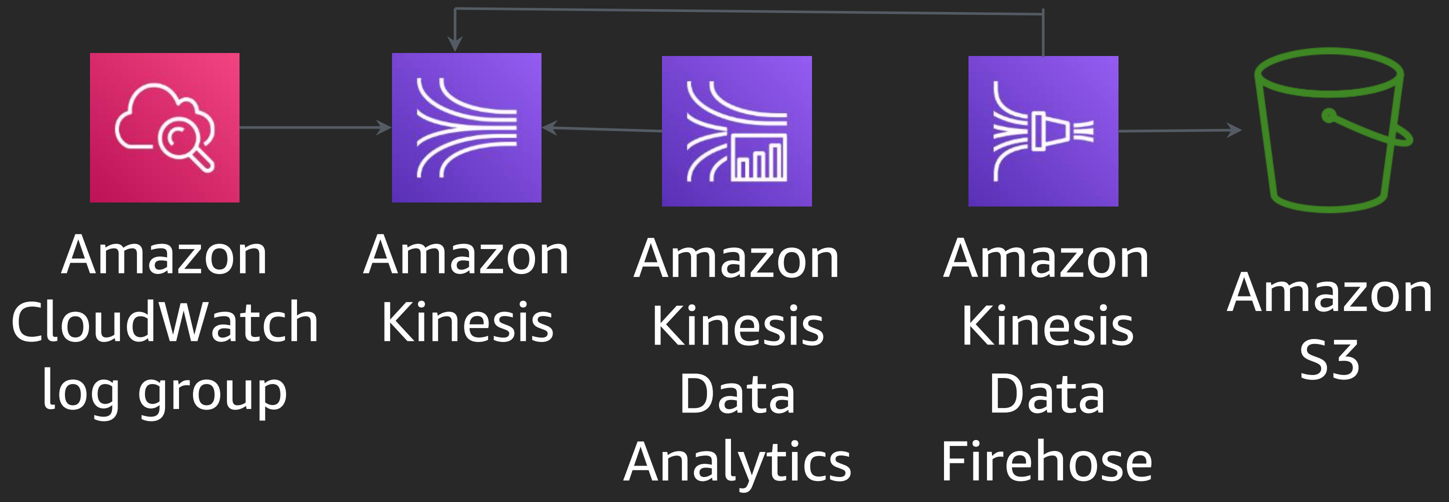


- Cloud platform log ingestion
- Cloud compliance monitoring
- ETL

Lessons learned

- Funnel to singular Amazon SQS/Lambda
- Cross-account roles need to be carefully managed
- Log grokking and alerting at hyper scale
- Image optimization is free; no more build time requirements

Event-based IRL: Data processing (1)


Amazon CloudWatch log group
Amazon Kinesis
Amazon Kinesis Data Analytics
Amazon Kinesis Data Firehose
Amazon S3

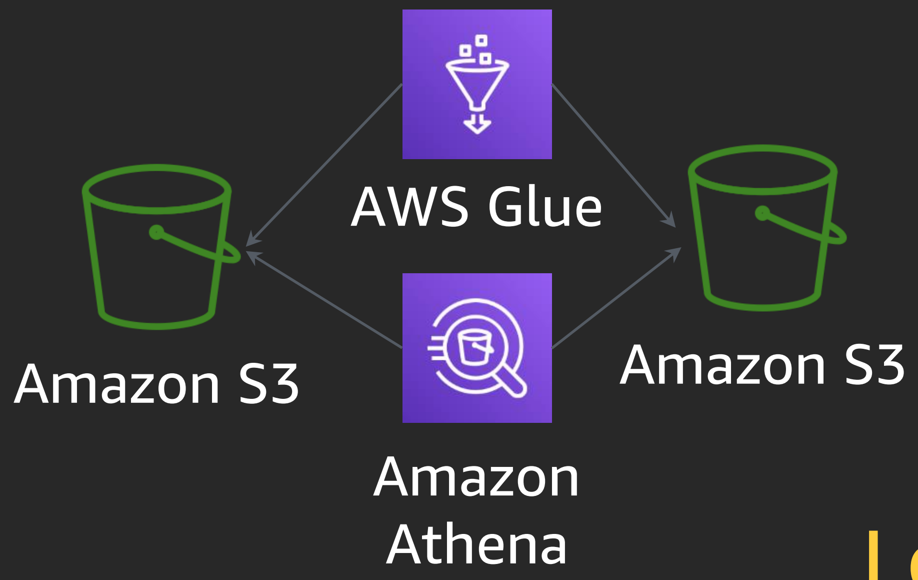
- Data loss prevention
- Monitoring -> ML features

Lessons learned

- Sharding vs. concurrency
- Amazon Kinesis SDK updates can be a surprise

- Multi-headed subscription does not exist currently

Event-based IRL: Data processing (2)

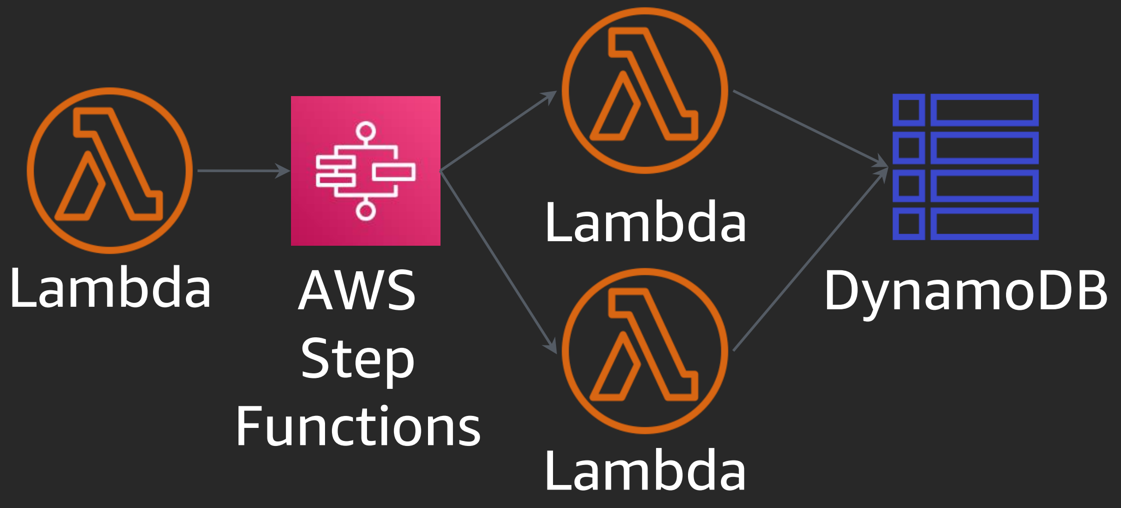


- Data catalog
- Auto-classification

Lessons learned

- Amazon Athena performance varies based on query complexity and dataset structuring
- .CSV file formats can be hard to be consistent
- Can't control "right" association of AWS Glue crawler IAM role
- Cross-account access can be a limiter

Event-based IRL: Rules engine

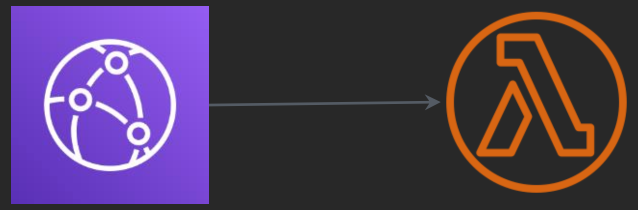


- Self-healing
- Policy-driven automation

Lessons learned

- Max input/result data size can be breached if a loop of wait conditions is implemented
- Redundant state outputs should use override logic
- Transmitting state data has helped using persistent store
- Cross-account Lambda invocation limit needs to be handled

Event-based IRL: CDN customization



Amazon CloudFront

Lambda@Edge

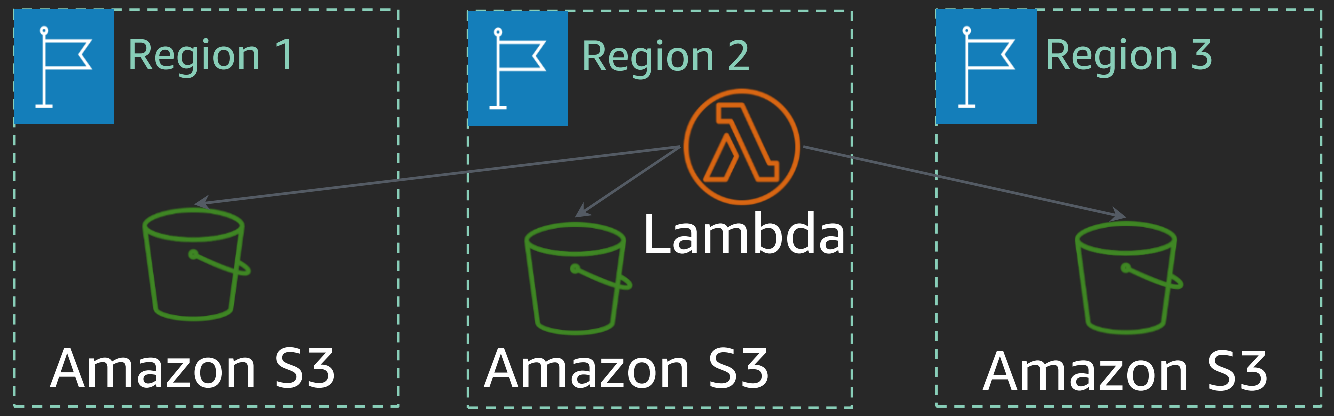
- Filter
- Rewrite

Lessons learned

- Lambda@Edge is only available in UE1 for config
- Logs are written to the same region as execution

- A/B testing made easy
- Routing is complex multi-region (failover conditions)

Event-based IRL: Multi-region resiliency

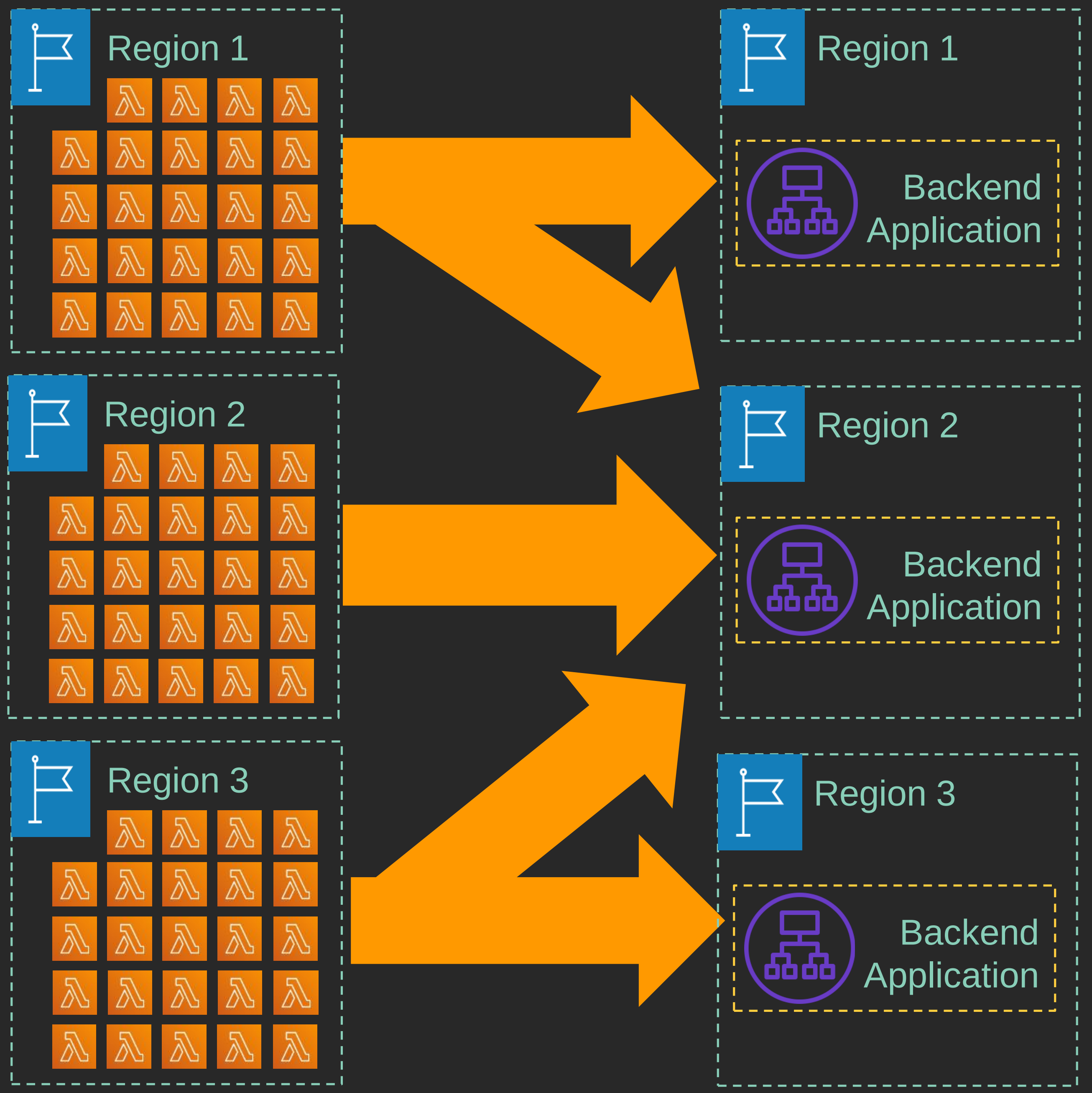


- Multi-write
- Active-active

Lessons learned

- AWS doesn't have a replication SLA
- Resiliency by reducing blast radius
- Allows rolling deployments
- Lambda@Edge can help with failover

Successful scaling can break the things



- Rapid scaling causes downstream impact
  - Downstream must support scaling
  - API gateways can be overrun
- Limits and enablement
  - Does downstream support rate limits?
  - How does the calling application handle rate limits?
  - What happens when you hit all the circuit breakers?

Lessons learned regarding security

- Lambda is ephemeral
  - Security focus shifts to surrounding infrastructure
  - Re-run containers can still be attacked
  - Memory space needs to be secured
- IAM can be complex
  - Conditional based on ARN
  - Trade-off roles vs. resource-based policy based on complexity
  - Managing ingress is better than managing egress
- Private network options have trade-offs
  - Public API endpoints
  - Private link
  - Private DNS resolution

Conclusion

The picture can't be displayed.

In summary

- Developing serverless solutions requires an AWS account – no true local development
- Need to optimize deployment architecture
- Need to enable observability from the onset

Thank you!

The picture can't be displayed.

Please complete the session survey in the mobile app.

The picture can't be displayed.