AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

ADM303

AWS ADVERTISING AND MARKETING TECHNOLOGY

Best practices for cloud-based real-time ad platforms

Gerry Louw
Global Head of Technology - Advertising and Marketing Technology
Amazon Web Services

Akhil Aendapally
Senior Solutions Architect
Amazon Web Services

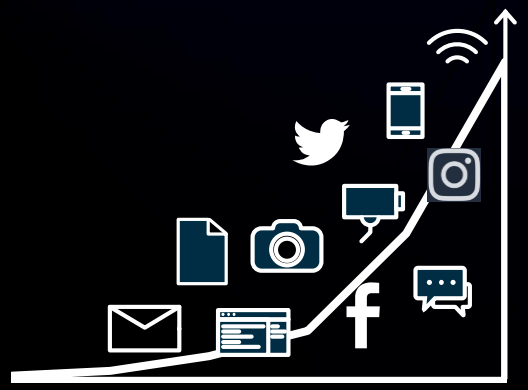
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Survey

- Volume:
- < 500K QPS
- < 1MM QPS
- < 5 MM QPS
- < 50 MM QPS
- > 50 MM QPS

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Programmatic advertising challenges



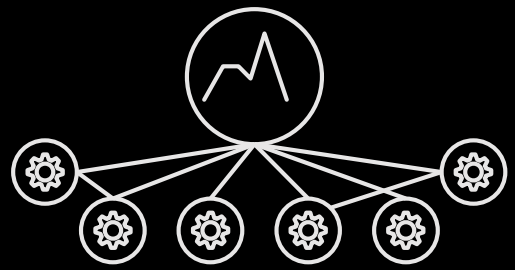
Margin and cost pressure

< $80 per billion bid requests



Millisecond latency response times

< 20 milliseconds



Variable global internet scale impression volumes

- **Billons of bids/requests per day**
- **Rapid change**

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

< 300 milliseconds

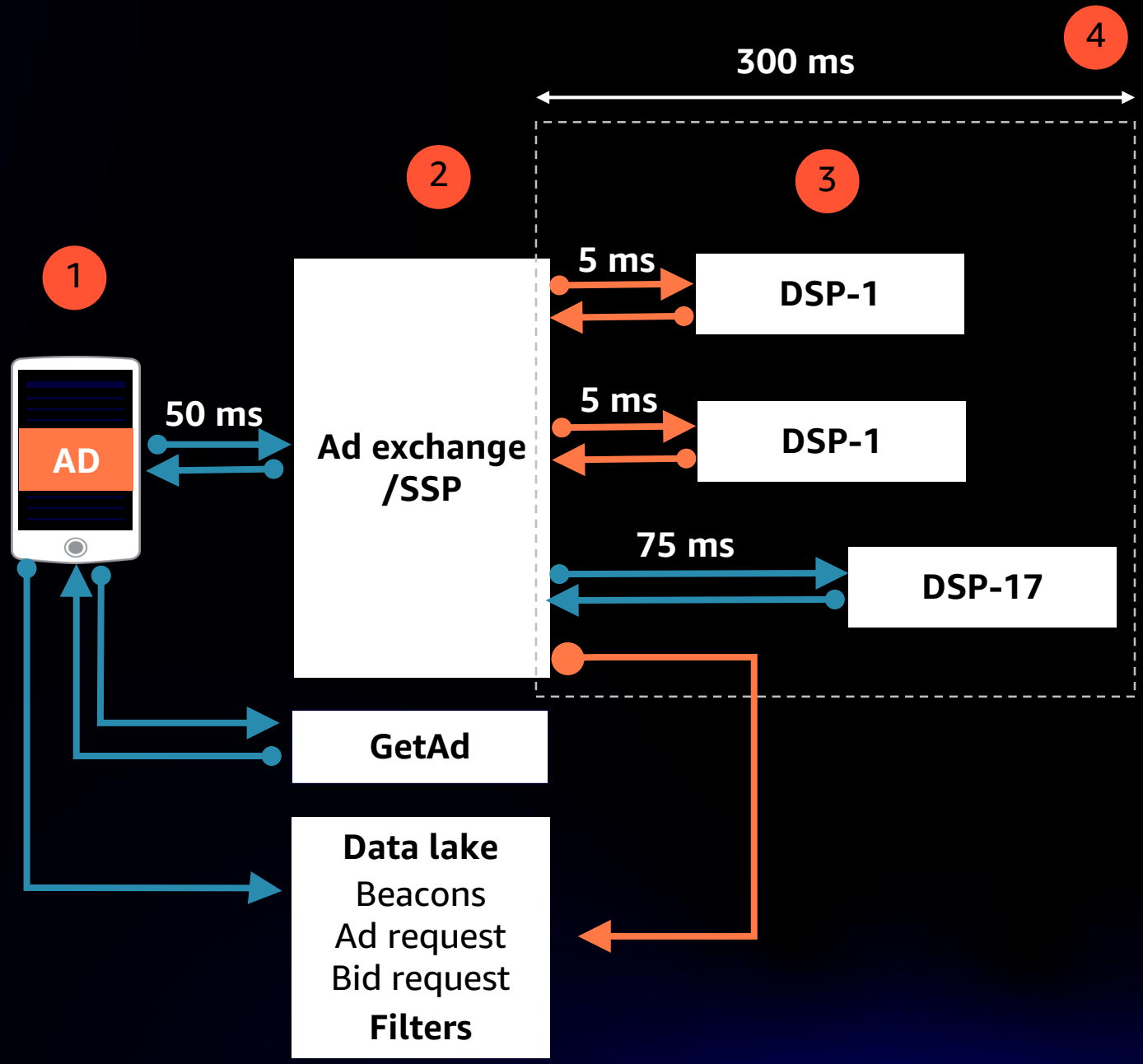
1. User opens web page, web page generates ad request (AR)

2. Supply Side Platform (SSP)/ad exchange identifies Demand Side Platforms (DSPs) that may be interested in the impression, and launches an auction

   The auction sends bid requests (BRs) to all DSPs

3. DSPs hosted inside AWS receive BR in 5 ms, network latency & distance increase time for DSPs outside AWS to receive the BR

4. Auction must complete within 300 ms



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Volume: For an SSP

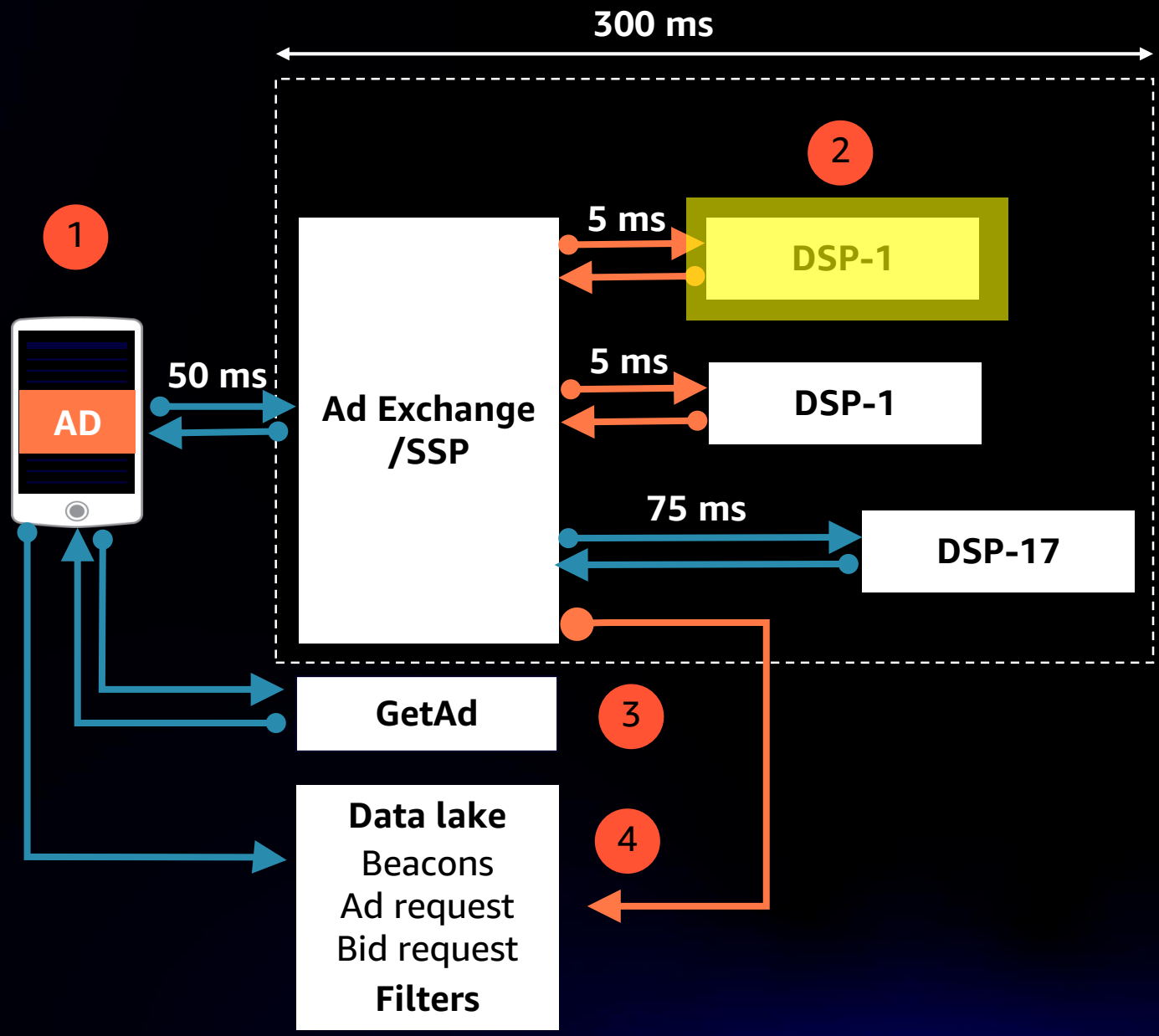
1. 1 ad request generates:

2. 17 bid requests
   17 bid requests responses
   1 ad request response

3. 1 GetAd request
   1 GetAd request response

4. 4 data beacons

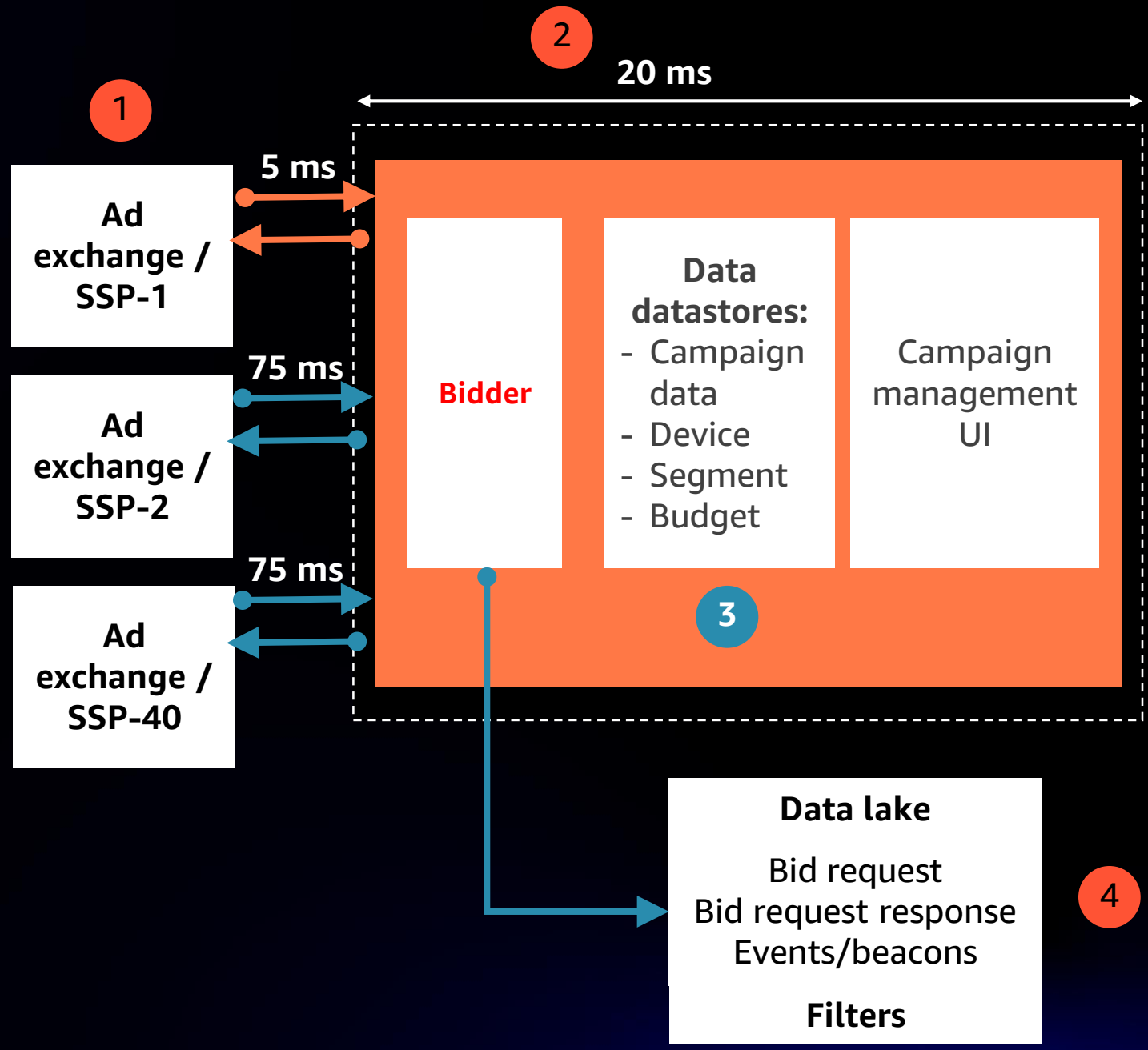
One ad request = 42 events captured



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

DSP: The bidder

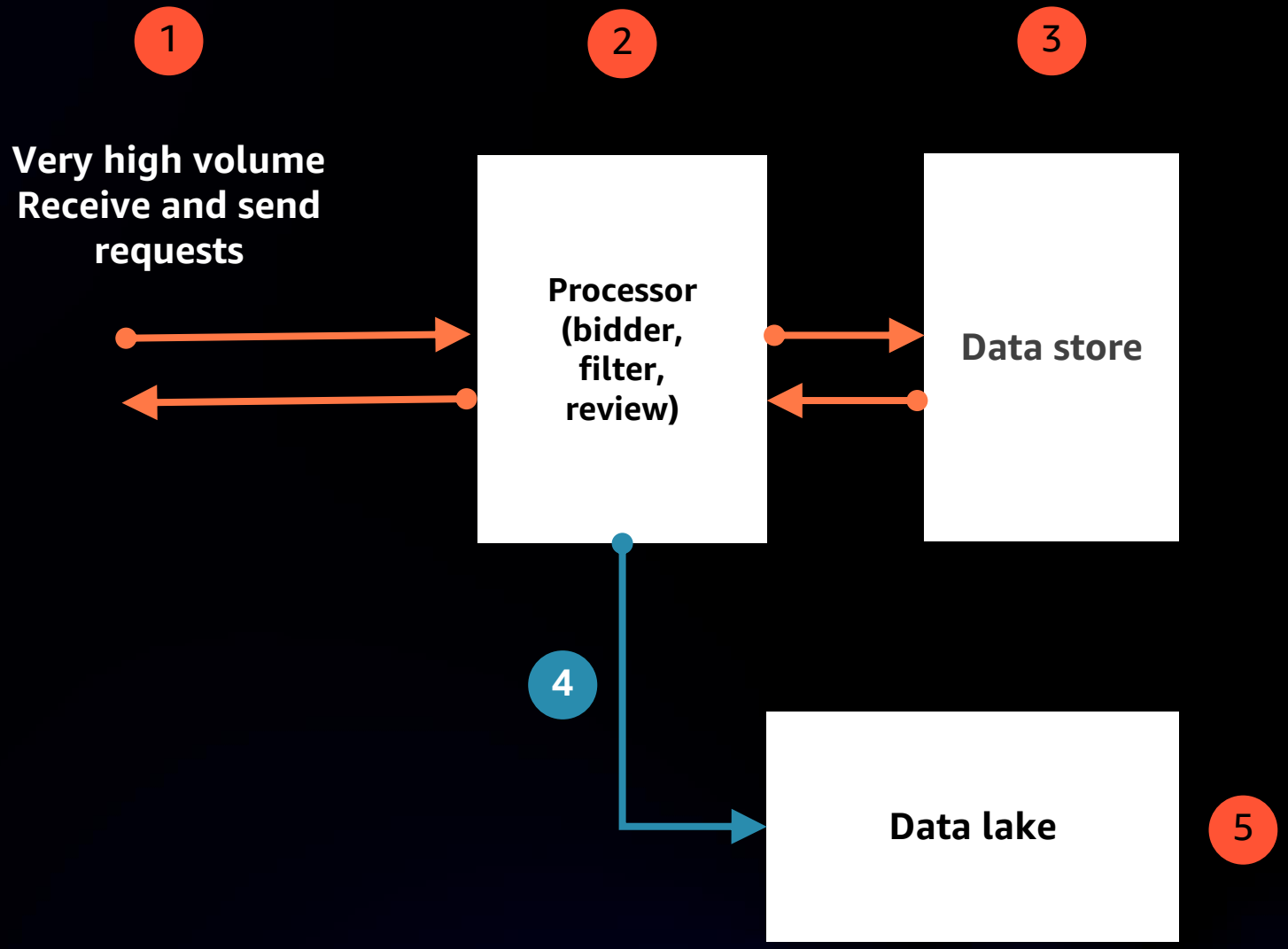
1. DSP receives bid requests from multiple ad exchanges
   - 50 k QPS per ad exchange
   - 100 k QPS to 2+MM QPS
2. Bidder response < 20 ms
3. Data stores:
   - 5 Bn devices
   - 1 MM Campaigns
   - < 2 ms response at 99th percentile
4. Store all requests, responses, and events



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Bidder pattern

1. Processor: receive and send requests, very high volume, < 300 ms roundtrip
2. Processor: parse request, retrieve additional data, evaluate, and respond
3. Data store: low latency read/write datastore
   - Cached data layer at edge
   - < 20 ms latency required
4. Data pipeline
   - Write transactional data lake
5. Data lake: persist transactional/log data



4 data beacons

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Best practices for a cloud-based real-time ad platform

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

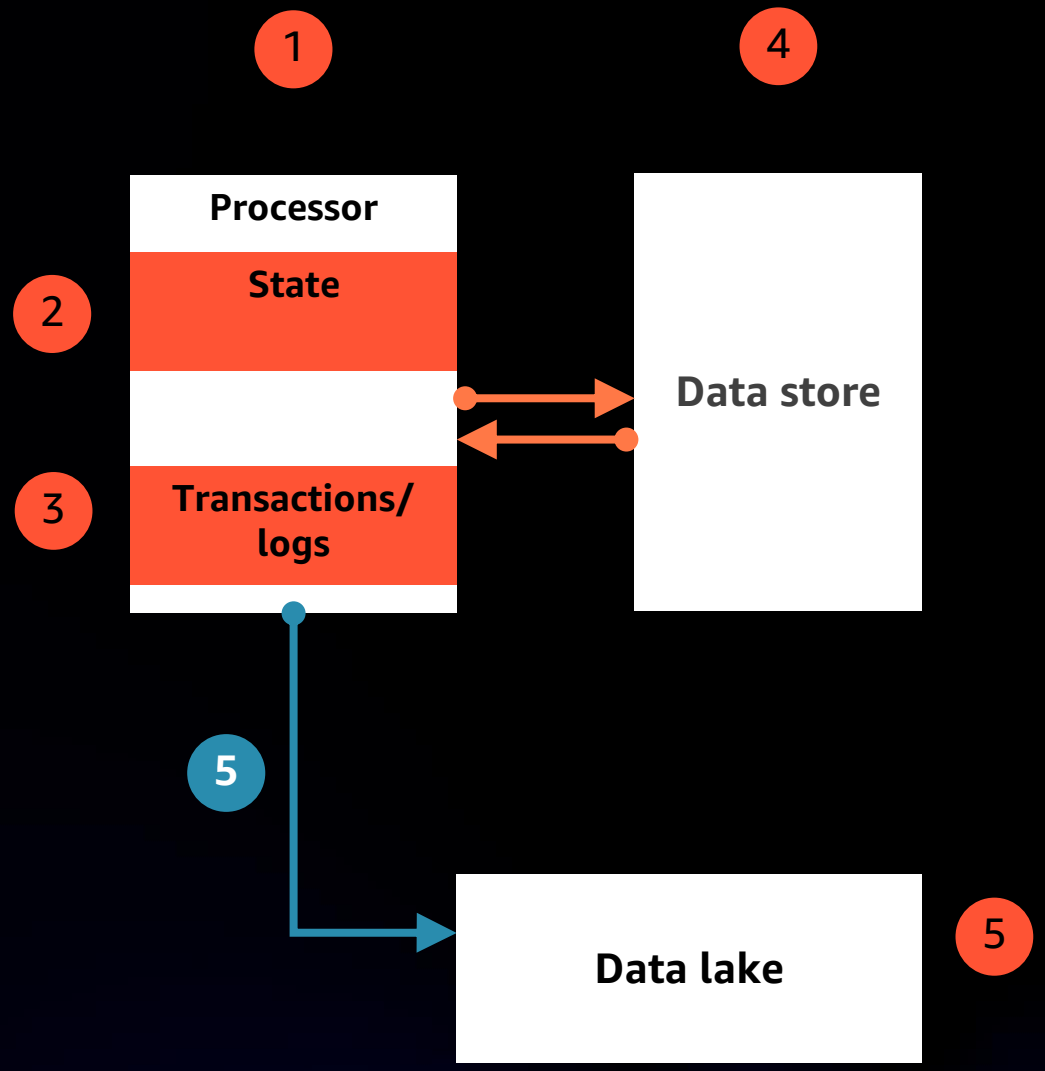
Processor – Best practice 1: Stateless architecture and auto scaling

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Stateful

1. Processor: deployment of a processor on an Amazon EC2 instance
2. State: select tables in memory, periodically refreshed/written to data stores
3. Transactions/logs: written to local disk
4. Data store: retain large, non-cached tables
5. Batch job: periodic batch job writes transactions/logs to data lake

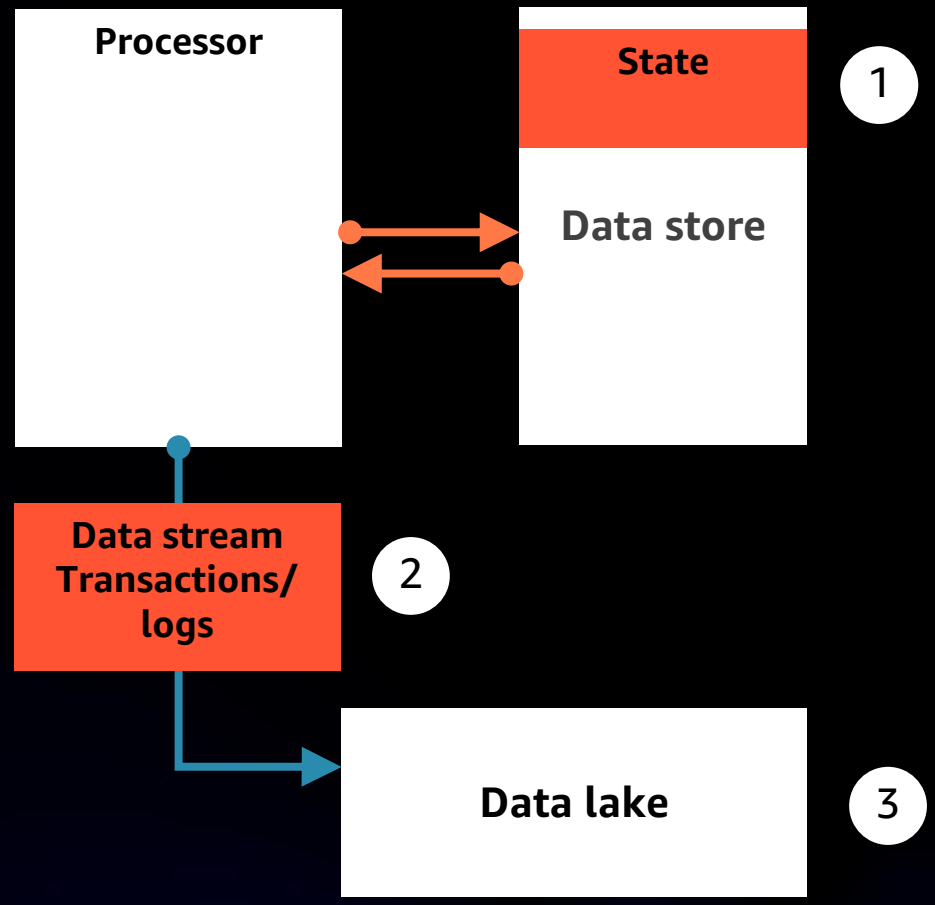
   Batch frequency = time needed to terminate an instance
6. Data lake: logs written to Amazon S3



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Stateless in three steps

1. Move state to data store
2. Replace batch job with data stream
3. Write data stream into data lake



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Processor – Best practice 2: Spot Instances and auto scaling

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Run production workloads on Spot Instances using Amazon EC2 Auto Scaling groups



Spot infrastructure

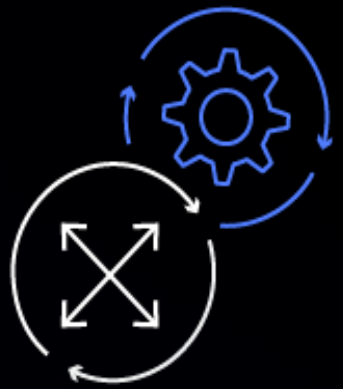
Same as on-demand and RIs

Up to 90 percent off



Spot pricing

Smooth, infrequent changes, no spikes, more predictable



Interruptions

Happen when EC2 needs capacity



Diversify

Choose different instance types, size, and AZ in a single fleet

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Spot Instances for Real-time bidding (RTB)

Best practice: minimize boot time of the bidder

- Perform all library installations and configurations in the AMI

Best practice: Use Amazon EC2 Auto Scaling groups to launch EC2 Instances

- Use the capacity optimized allocation strategy to reduce the possibility of interruptions
- Use Instance scaling policies, warm-up settings, and Instance weighting to fine tune capacity and costs

Lowest cost practice: 100 percent Spot Instances

- Target 100 percent of your non-database workload on Spot Instances
- Plan auto scaling to include on-demand as last resort in ASG

Lowest cost practice: Single Availability Zone deployment

- **Not a best practice,** lowest cost – deploy in **one AZ** only
- Have AMI and deployments scripts in another AZ/region as automated fail-over

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Results from using Auto Scaling and Spot Instances

Quantcast

Saved 60 percent on its QCLearn ML system and 25 percent for RTB using Amazon EC2 Spot Instances; scales up to 1,000 instances every day

smaato

61 percent cost savings on compute for real-time ad bidding using Amazon EC2 Spot Instances

gumgum

62 percent savings on real-time bidding and ML with Amazon EC2 Spot Instances

IPONWEB

80 percent utilization of Spot Instances on top of Auto Scaling groups; Spot Instances enabled IPONWEB to grow infrastructure on AWS by about 15 percent without incurring additional costs

AppsFlyer

Processes 100 billion daily events for fraud detection and prevention, saving 75–80 percent using Amazon EC2 Spot versus On-Demand Instances

BEESWAX

"If you care about operating costs, Spot is going to save you a ton of money." Ram Kumar Rengaswamy, co-founder and CTO of Beeswax

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Processor – Best practice 3: AWS Graviton2

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Graviton2-based instances

Up to **40 percent better price-performance** over comparable x86-based instances

M6g	T4g	C6g	R6g	C6gn	X2gd
General purpose workloads	Burstable general purpose workloads	Compute-intensive workloads	Memory-intensive workloads	100 Gbps networking 38 Gbps EBS	Memory optimized Lowest $/GiB of RAM

Local NVMe-based SSD storage options are also available:
general purpose (M6gd), compute-optimized (C6gd), and memory-optimized (R6gd & X2gd)

M, C, R, and X instance types also have bare-metal options:
(M6g.metal, M6gd.metal, C6g.metal, C6gd.metal, R6g.metal, R6gd.metal, X2gd.metal)

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Graviton2 ease of adoption

Difficulty	Workload	Actions
Virtually no effort	RDS, Aurora, ElastiCache, and Elasticsearch	Upgrade to latest and enjoy
Super easy	EMR	Typically, just works
Quite easy	Linux – Interpreted and JIT'd languages (e.g., Java, PHP, Node.js)	Select Arm64 AMI and Install Bonus if using containerized
More involved	Linux – Compiled languages (e.g., C/C++, Python, Go)	Select Arm64 AMI and compile + port any intrinsics or assembly
Some work, high reward	Microsoft Windows – .NET	Migrate to .NET core, run on Linux/arm64
Sorry, not yet	Microsoft Windows	Microsoft Windows Server not yet available for arm64

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Results from using AWS Graviton2



NextRoll: 50 percent reduction in compute costs for RTB using AWS Graviton2–based Amazon EC2 Instances, using M6g and C6g instances (Erlang, Go, and Java)



Netflix: 50 percent performance increase M5 to M6g (Java benchmark)



Nielsen “seamlessly transferred this Java application to Amazon EC2 A1 instances powered by the AWS Graviton processor. We’ve since tested the new AWS Graviton2-based M6g instances and it was able to handle twice the load of an A1” - Chris Nicotra, SVP Digital, at Nielsen.



Jampp processes more than 2 million ad bid requests per second from 20+ different sources across the globe. “On average, we experienced 15 percent better performance for our workloads and a total cost saving of about 30 percent considering the lower per-instance cost and the fewer number of instances due to better performance.” - Manickababu Muthu, CTO, Jampp

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Results from using Auto-Scaling, Spot Instances and AWS Graviton2

Current status	% Cost Reduction	Cost reduction	Total Cost remaining
Application is statefull			$100

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Results from using Auto-Scaling, Spot Instances and AWS Graviton2

Current status	% Cost Reduction	Cost reduction	Total Cost remaining
Application is statefull			$100
Move to stateless			
Implement auto-scaling	32%	$32	$68

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Results from using Auto-Scaling, Spot Instances and AWS Graviton2

Current status	% Cost Reduction	Cost reduction	Total Cost remaining
Application is statefull			$100
Move to stateless			
Implement auto-scaling	32%	$32	$68
On-demand to Spot Instances	60%	$41	$27

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Results from using Auto-Scaling, Spot Instances and AWS Graviton2

Current status	% Cost Reduction	Cost reduction	Total Cost remaining
Application is statefull			$100
Move to stateless			
Implement auto-scaling	32%	$32	$68
On-demand to Spot Instances	60%	$41	$27
Port to Graviton2	40%	$10.88	**$16.32**

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Results from using Auto-Scaling, Spot Instances and AWS Graviton2

Current status	% Cost Reduction	Cost reduction	Total Cost remaining
Application is statefull			$100
Move to stateless			
Implement auto-scaling	32%	$32	$68
On-demand to Spot Instances	60%	$41	$27
Port to Graviton2	40%	$10.88	**$16.32**

Total Cost reduction	**84%**

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Best practice 4: Stateless data pipeline and data streaming

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data pipeline

Best practice: migration to data streaming

- Decouple processor and data pipeline in move to stateless
- Define and budget for overlap period prior to start
- Define acceptable delta between batch/stream prior to start

Best practice: Amazon Kinesis or Amazon MSK

- Combine records to optimize 25KB PUT size
- Combine to reduce number of commits

Best practice: avoid data loss

- Define acceptable data loss in data pipeline, architect accordingly
- Use probability of spot termination to define risk of losing data

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Best practice 5: Database choice and optimization

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Database

Best practice: Aerospike on AWS

- Aerospike: Low cost, sub-millisecond latency at scale
  - Aerospike Database Enterprise Edition on AWS
  - Aerospike petabyte scale benchmark:

theTradeDesk

IAS Integral Ad Science

BEESWAX

Aerospike can deliver 4–5 million transactions per second (TPS) for read-only and mixed workloads with sub-millisecond latencies on a 20-node AWS cluster and more than 5 million TPS for read-only operations and 3.7 million TPS for an 80/20 mix of read/write operations with server-side latencies of less than 1 ms in nearly all cases

AEROSPIKE

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Database

Best practice: Database design

- Incremental record update vs full record update
- Design TTL for record
- Controlled width of table, and attribute TTL

Best practice: Data replication

- Regional data sets - avoid full data replication to all regions
- Allocate and consume regional budgets

Low cost practice: Lower regional data transfer costs

- Single AZ database deployment, snapshot to DR AZ/Region
- Keep traffic in AZ – make cluster “rack aware”
- Compress data before persistence (throughput and data transfer)

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Database

Best practice: careful instance selection

NVMe SSDs (optimize for specific workload):

- Compress and write large objects: CPU to disk ratio
- Read many smaller objects: DRAM to SSD

Network: Instance network throughput capacity

Use AWS Graviton2

LOWEST cost practice 12:

- Select AWS Graviton2 for your database/data processing
- Ask your database vendor ETA Arm-based version

AWS Graviton2 – results

- Amazon EMR and Spark: 30 percent lower cost, 15 percent improved performance
- PostgreSQL: M6g is between 12 percent and 46 percent faster and has up to 34 percent better response times versus M5.
- Apache Hadoop 3.3.0: first release to support ARM architectures
- Elasticsearch price/performance benefits of AWS Graviton2 (up to 25 percent better performance and 20 percent lower cost versus M5 instances)
- MongoDB: 117 percent higher throughput and 20 percent lower cost on R6g versus R5 instances

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Best practice 6: Reducing data transfer costs

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Reducing data transfer costs

Best practices:

- Seek private pricing for AWS Data Transfer and AWS Direct Connect if you have significant transaction volumes out to partners
- Explore AWS PrivateLink for same Region connectivity to customers/partners
- Reduce data transfer out by predicting duplicate bids/requests (traffic filtering)
- Reduce the size of records being transmitted (compress, value by reference, remove redundant data)
- Reduce frequency of database replication across Regions and AZs
- Find other ad techs on AWS and co-locate in the same Region/AZs (data transfer between partners when both are inside AWS is a much lower cost)

aws

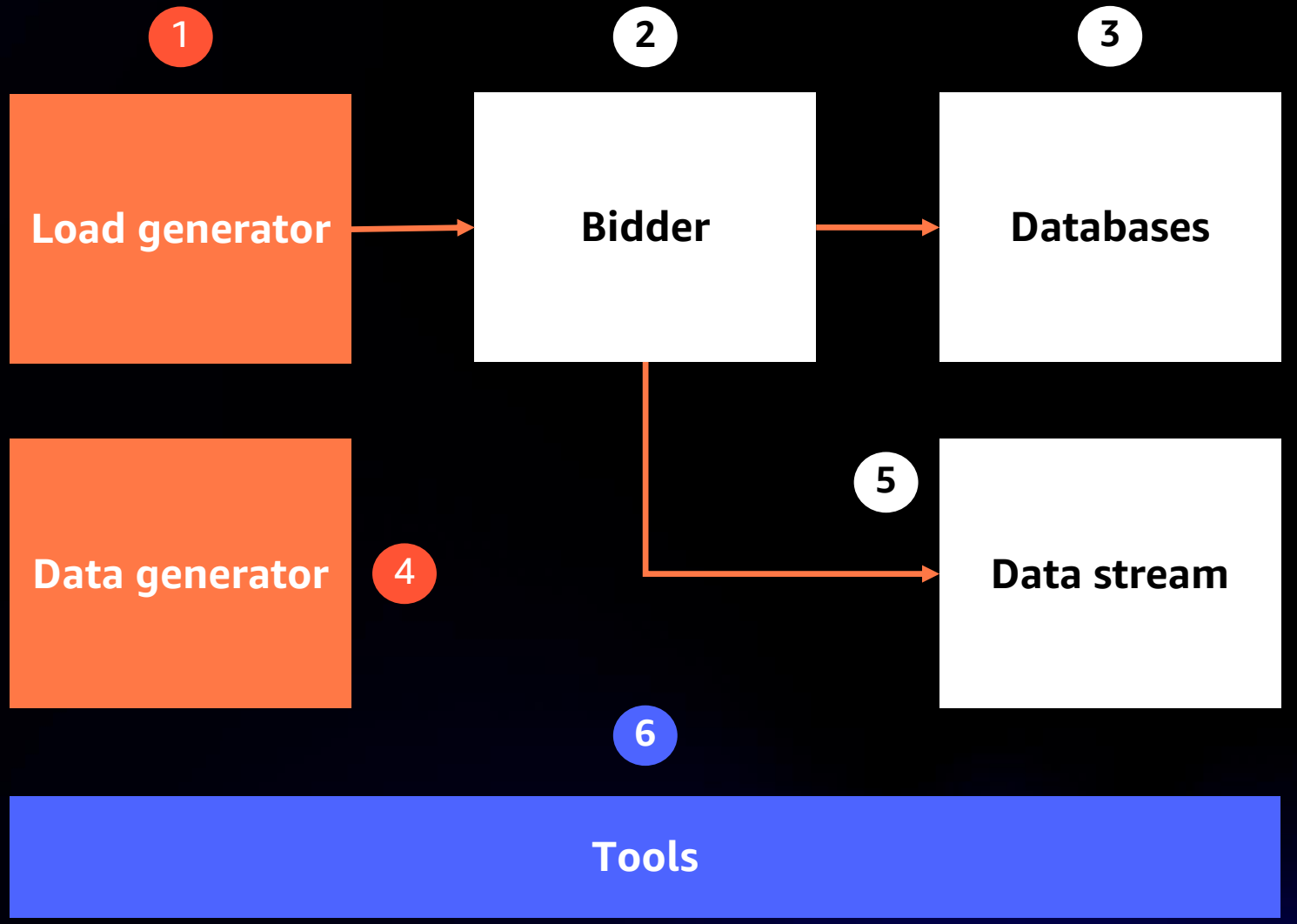
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Getting starting quickly: Real-time bidding on AWS Solutions

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Real-time bidding on AWS Solutions

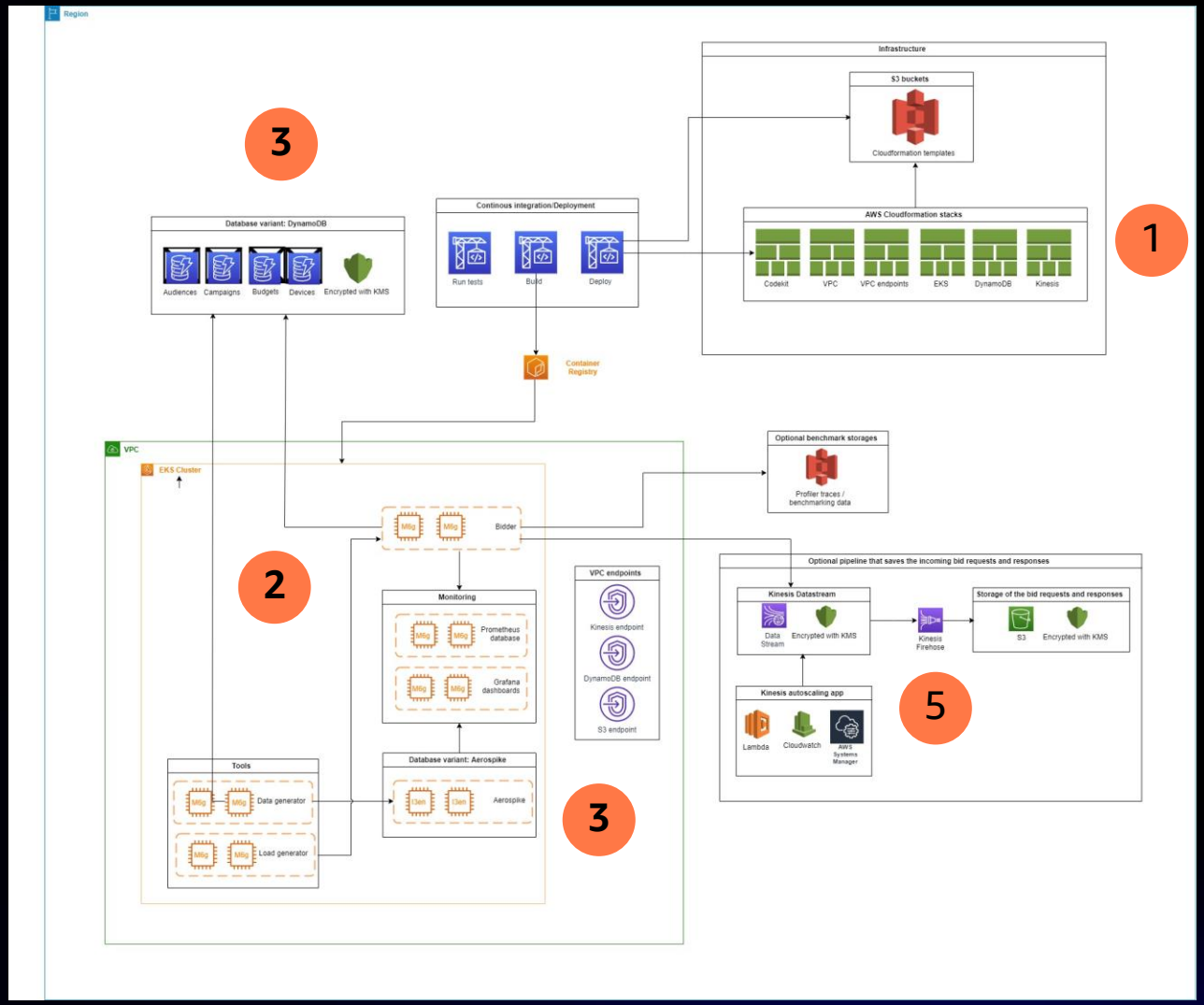
1. **Data generator:** Generate bid requests to stress test bidder
2. **Bidder:** Receive bid requests, retrieve data needed to evaluate, select best campaign, price bid, and respond with bid
3. **Databases:** Device profiles, campaigns, segments and budgets
4. **Data generator:** Creates artificial data to populate device profiles, campaigns, segments, and budgets databases.
5. **Data stream:** Amazon Kinesis write log entries to Amazon S3
6. **Tools:** Code deployment, monitoring, and metrics visualization tools



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Real-time bidding on AWS Solutions

1. **AWS CloudFormation:** Modular deployment, stateless architecture
2. **Containers:** Discrete container per module, and Spot Instances
3. **Databases:** Amazon DynamoDB or Aerospike
4. **AWS Graviton2:** Use AWS Graviton2 where possible for compute
5. **Data streaming:** Kinesis or Amazon MSK



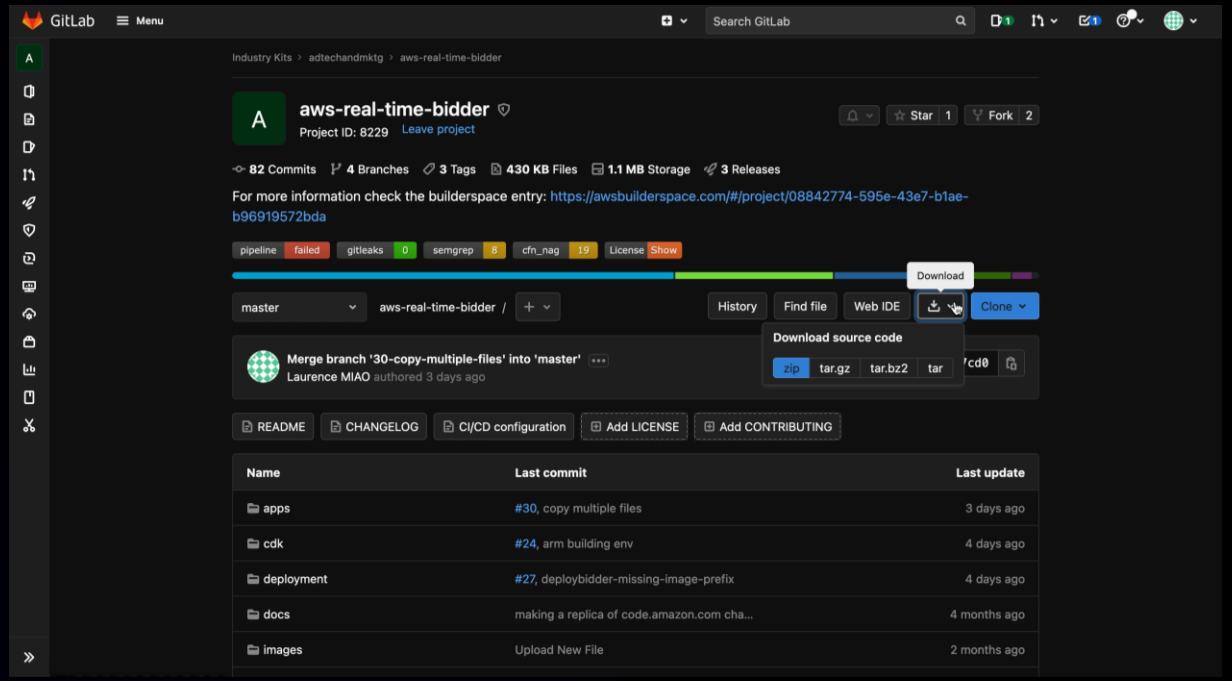
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Real-time bidding on AWS Solutions: Deployment

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Deployments steps

1. Download Source Code
2. Provide environment variables
3. Create pipeline using AWS CDK
4. Push source code to AWS CodeCommit
5. Deploy Infrastructure
6. Connect to Amazon EKS Cluster
7. Run benchmarks



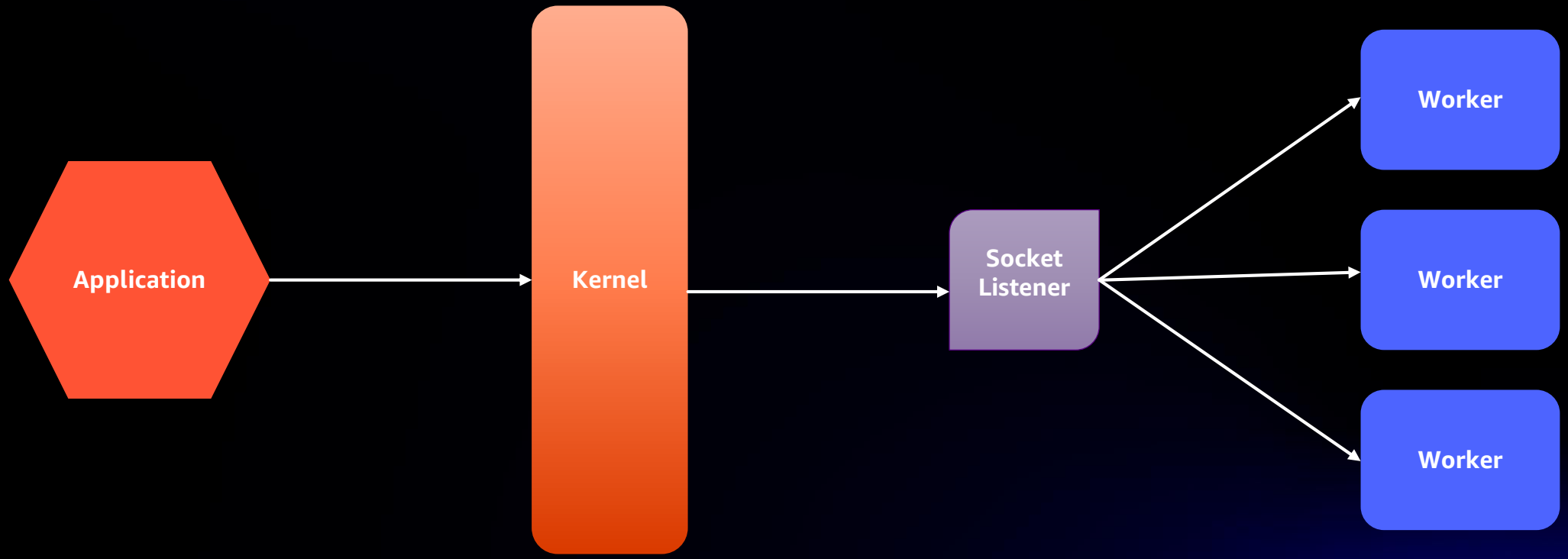
https://awsbuilderspace.com/#/project/08842774-595e-43e7-b1ae-b96919572bda

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Real-time bidding on AWS Solutions: Lessons learned

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

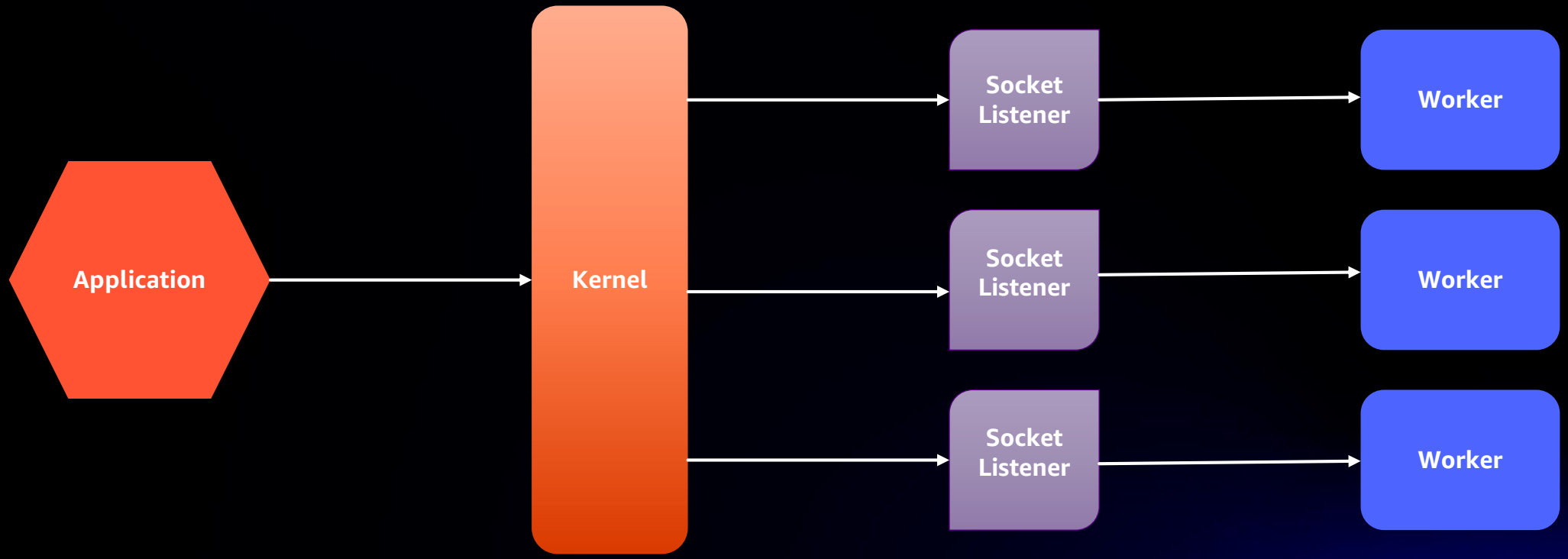
Low latency: #1 enable SO_REUSEPORT



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Low latency: #1 enable SO_REUSEPORT

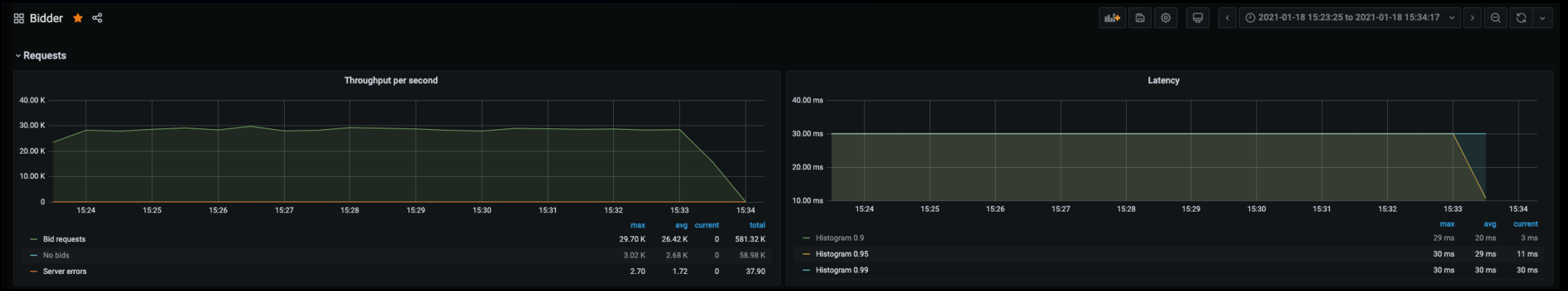
- With SO_REUSEPORT option enabled there are multiple socket listeners for each IP address and port combination, one for each worker process.



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Low latency 2: Optimize Instances per CPU

1 bidder on 64 CPUs: 26.4K / 0.99 = 30 ms



32 bidders, each ONE on 2 CPUs: 100.27K / 0.99 = 12 ms



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Low latency 3: Use FastJSON

Parsing a bid request takes 14 percent of CPU load

Using Standard JSON: 100.27K / 0.99 = 12 ms



Using FastJSON: 124.02K/ 0.99 = 6 ms



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Low latency 4: Run fastHTTP

Using net_HTTP: 147.33K / 0.99 = 5 ms



Using fastHTTP: 176.91K/ 0.99 = 990 µs



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

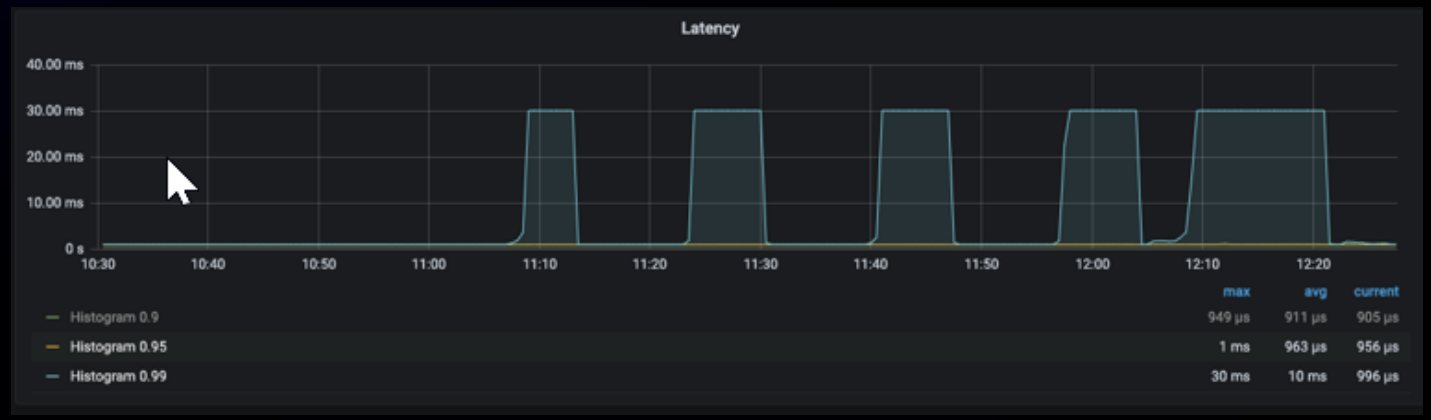
Low latency 5: Non-blocking data stream

- Make data streaming a non-blocking event
- Monitor the depth of your aggregation queue
- Ensure latency of data stream does not exceed latency of data retrieval plus ML inference
- Write bid request and bid request response together

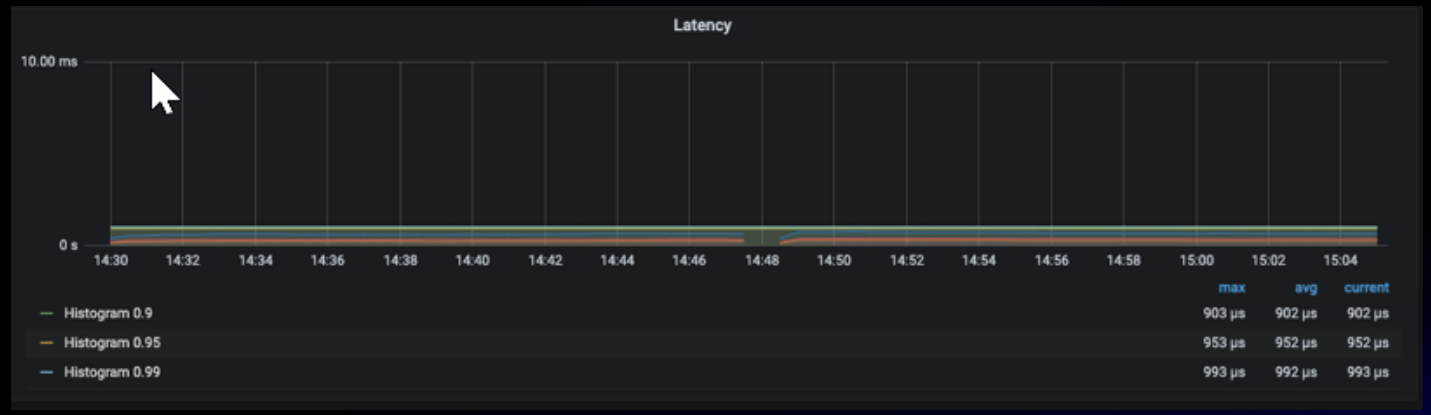
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Low latency 6: Use Zstandard compression

Using standard Kinesis producer: 10 ms



Using Kinesis_zstd: = 992 µs



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Low latency 7: Optimize Prometheus summaries

Using standard Prometheus summaries: 359.91K / 50 ms



Using modified Prometheus summaries: 383.91 / 228 µs



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What are the results

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Benchmark results – Bidder QPS and latency

Experiment Results

Baseline Instance: One **m6g.16 xl**, 64 vCPUs
CPU Utilization 70%

BIDDER	MAX QPS (in thousands)	Latency at 99th percentile (in milliseconds)
baseline	**7.48**	**30**
SO_REUSEPORT = ON	29.7	30
optimize instance count (1 Bidder on 64 vCPUs)	29.7	30
optimize instance count (32 bidder on 2 vCPUs)	100.27	12
use standard JSON	100.27	12
use FastJSON	124.02	6
use net_HTTP	147.33	5
use fastHTTP	**176.91**	**0.99**
TOTAL IMPROVEMENT	**169.43**	**29.01**
% Improvement	**2165%**	**2930%**

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Gerry Louw
louwgerr@amazon.com

Akhil Aendapally
aaedapa@amazon.com

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.