



Being a good citizen in an event driven world

Ajay Nair

Principal Product Manager

Amazon Web Services



“Serverless will fundamentally change how we build business around technology and how you code.” -
Geekwire

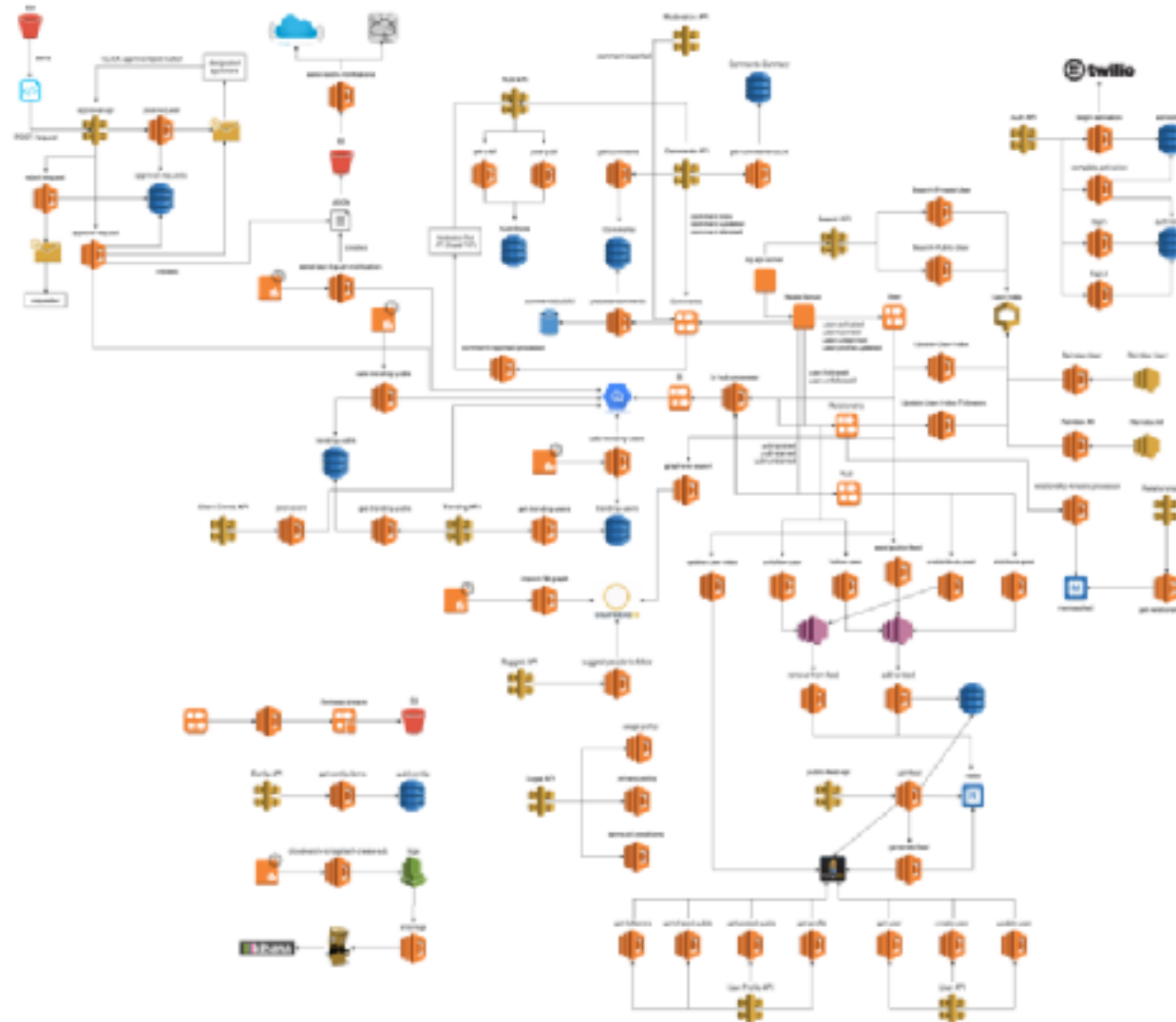
TODAY



(from "AWS Lambda from the trenches" by Yan Cui)

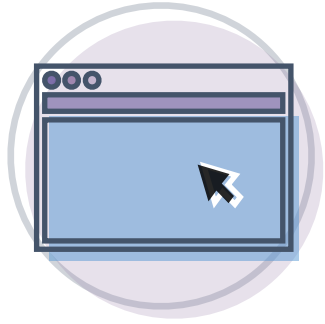
@ajaynairthinks

TOMORROW



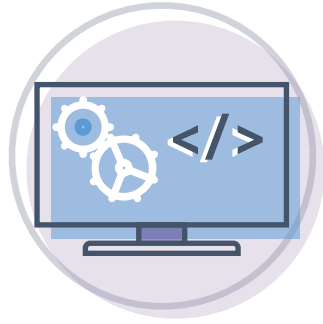
(from "AWS Lambda from the trenches" by Yan Cui @ajaynairthinks)

USE CASES



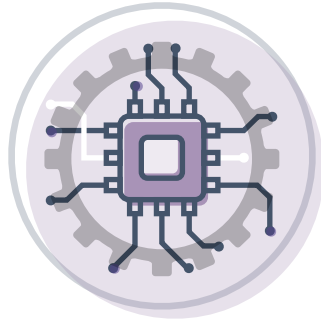
Web Applications

- Static websites
- Complex web apps



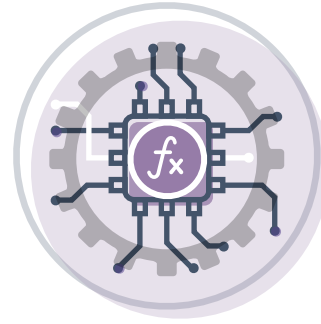
Backends

- Apps & services
- Mobile
- IoT



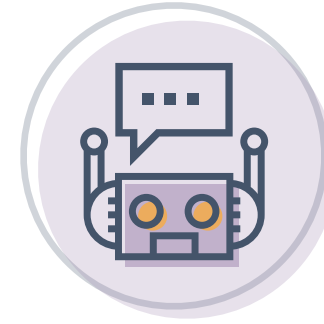
Data Processing

- Real-time data
- Streaming data
- Media content



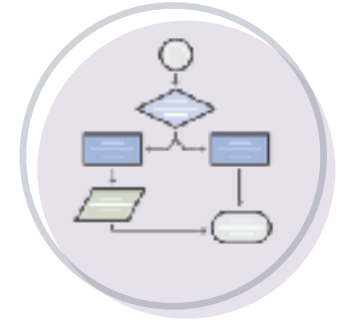
Big Data

- MapReduce
- Batch



Chatbots

- Powering chatbot logic
- Powering voice-enabled apps
- Alexa Skills Kit

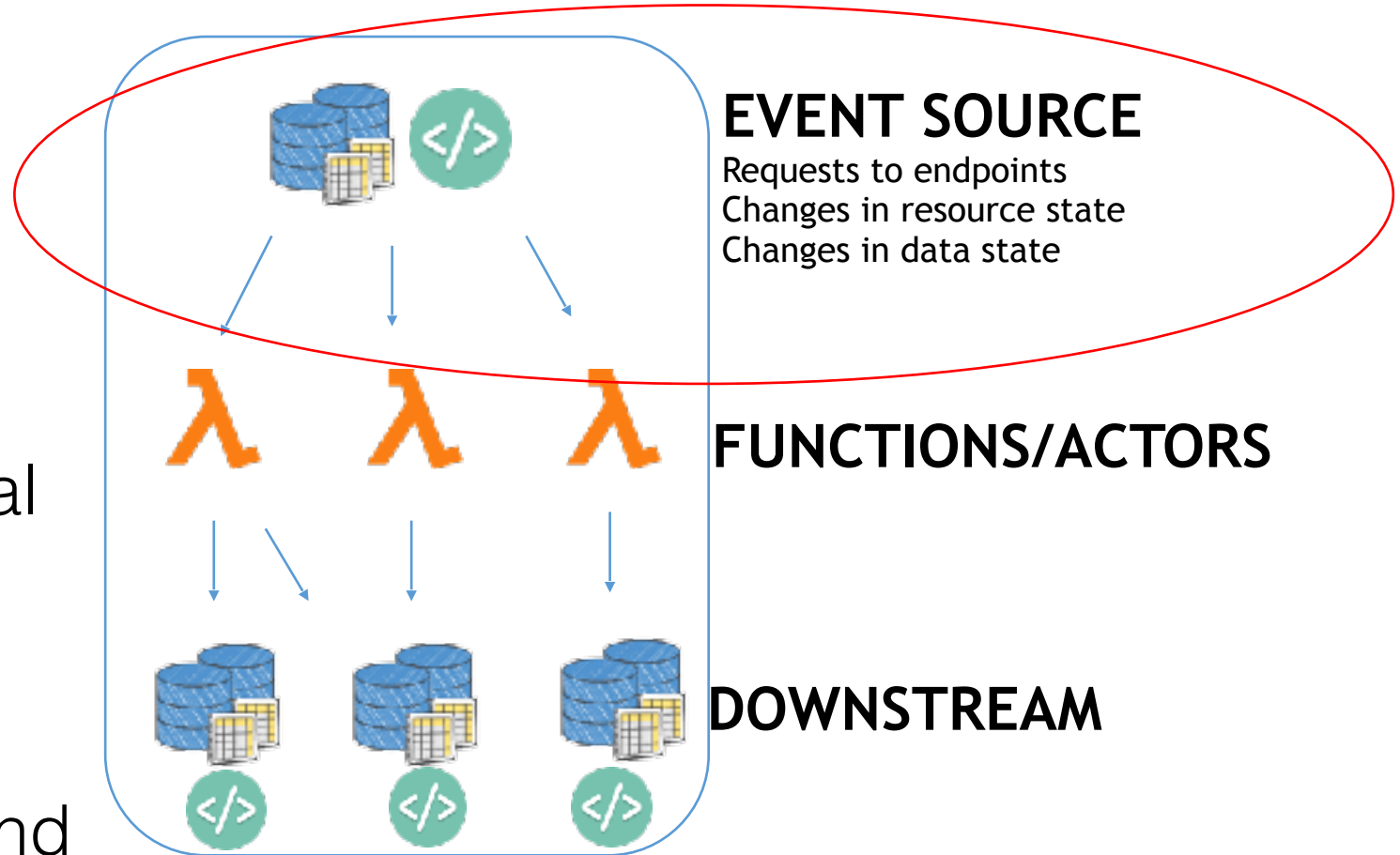


Autonomous IT

- Policy engines
- Infrastructure management

EVENT DRIVEN ARCHITECTURES (GROSSLY SIMPLIFIED)

- ✓ Communicate through events and APIs
- ✓ Stateless, ephemeral actors
- ✓ Separation of logic from data, cache, and state

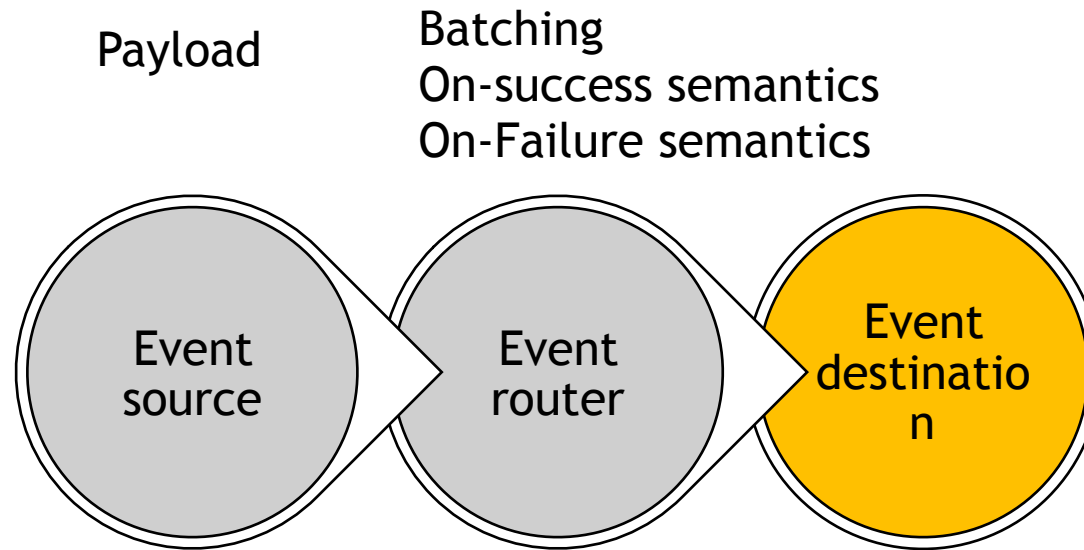




How do I build an effective event source?

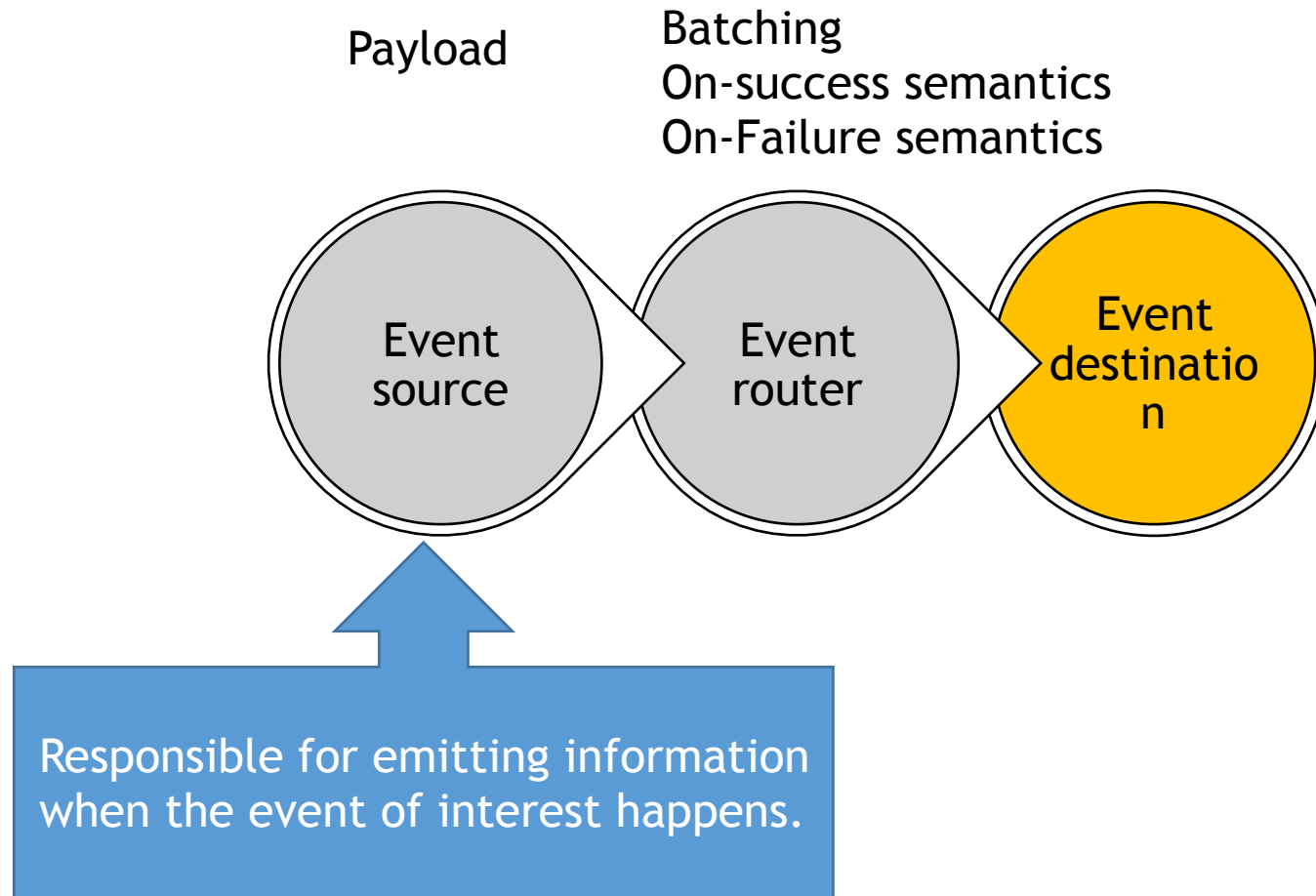
Event delivery concepts

EVENT DELIVERY CONCEPTS

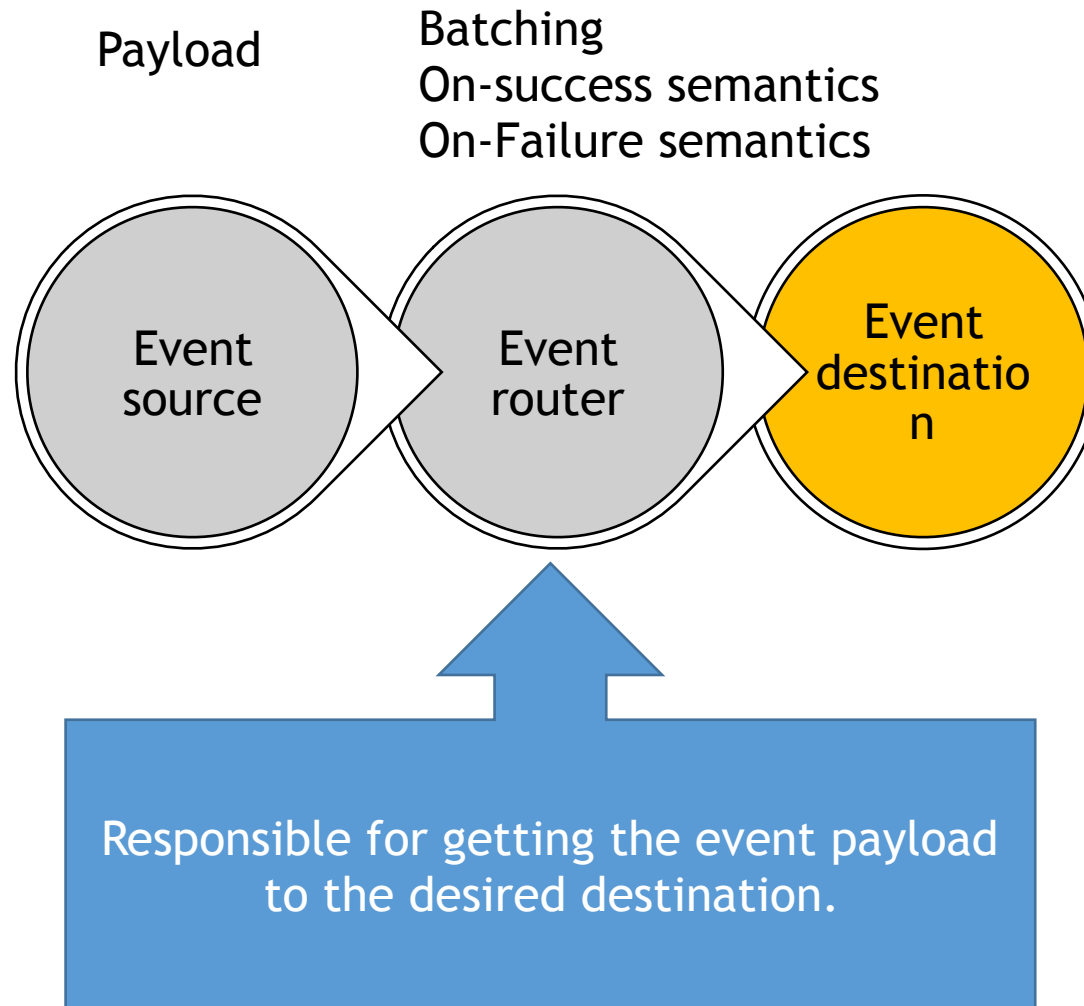


“An event is a signal emitted by a component upon reaching a given state.” – Reactive Manifesto









EVENT DELIVERY CONCEPTS



EVENT DELIVERY CONCEPTS



EVENT DELIVERY CONCEPTS

	Event source	Event router
<i>Controls...</i>	<i>Payload</i> <i>Retention</i>	<i>On-success semantics</i> <i>Filtering</i> <i>On-Failure semantics</i>
AWS Examples	 Amazon DynamoDB  Amazon S3  CloudWatch events (CRON)  EC2	 Amazon SNS  CloudWatch events Rules  Lambda (Event Source Mappings)  AWS IoT (Gateway)

DECISION #1 - PAYLOAD

WHAT SHOULD BE IN THE EVENT?

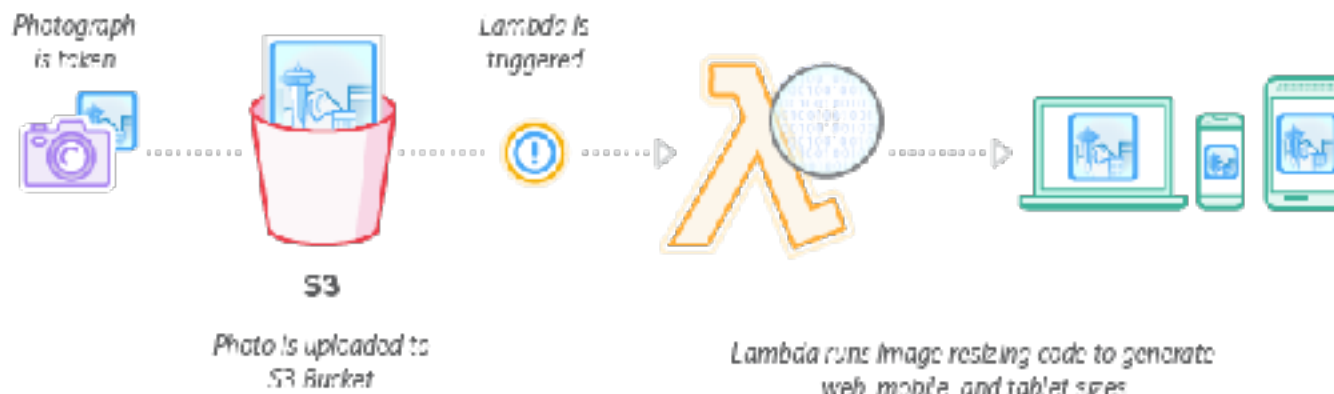
- **Baseline - Provenance**

- “What happened for this notification to occur?”

Pattern #1 - Event notification pattern*

- Event processor expected to contact event source to do work
- Requires *characteristics/identifier* - “What can you tell the event destination about the event source and entity affected?”

Example: Image Thumbnail Creation



@ajaynairthinks

* Martin Fowler - “What do you mean by “Event-Driven”?” - Feb 2011

WHAT SHOULD BE IN THE EVENT?

- **Baseline - Provenance**

- “What happened for this notification to occur?”

Pattern #2 - Event-carried state transfer pattern*

- Event processor not expected to contact event source
- Requires *payload* i.e. state you want to transfer downstream

Example: Sensors in Tractor Detect Need for a Spare Part and Automatically Place Order



EXAMPLE EVENTS

S3

```
{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "The time, in ISO-8601 format, for example, 1970-01-01T00:00:00.000Z, when S3 finished processing the request.",
      "eventName": "event-type",
      "userIdentity": {
        "principalId": "Amazon-customer-ID-of-the-user-who-caused-the-event"
      },
      "requestParameters": {
        "sourceIPAddress": "ip-address-where-request-came-from"
      },
      "responseElements": {
        "x-amz-request-id": "Amazon S3 generated request ID",
        "x-amz-id-2": "Amazon S3 host that processed the request"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "ID found in the bucket notification configuration",
        "bucket": {
          "name": "bucket-name",
          "ownerIdentity": {
            "principalId": "Amazon-customer-ID-of-the-bucket-owner"
          },
          "arn": "bucket-ARN"
        },
        "object": {
          "key": "object-key",
          "size": "object-size",
          "eTag": "object eTag",
          "versionId": "object version if bucket is versioning-enabled, otherwise null",
          "sequencer": "a string representation of a hexadecimal value used to uniquely identify the object, only used with PUTs and DELETES"
        }
      }
    }
  ]
}
```

Provenance

Payload

Characteristics

IoT data through Kinesis

```
"Records": [
  {
    "eventID": "shardId-000000000000:49545115243490985018280067714973144582180062593244200961",
    "eventVersion": "1.0",
    "kinesis": {
      "partitionKey": "partitionKey-3",
      "data": "SGVsbG8sIHRoaXMgaXMgYSB0ZXN0IDEyMy4=",
      "kinesisSchemaVersion": "1.0",
      "sequenceNumber": "49545115243490985018280067714973144582180062593244200961"
    },
    "invokeldentityArn": "identityarn",
    "eventName": "aws:kinesis:record",
    "eventSourceARN": "eventsourcern",
    "eventSource": "aws:kinesis",
    "awsRegion": "us-east-1"
  }
]
```


DECISION #2 – EVENT STORE

OPTIONAL: EVENT STORE

Durability

Events are accessible even if event source is down

Cost

Additional storage, data transfer, and service costs

VS.

Retention

Events can be revisited until processed

Complexity

New service dependency, operational surface area

Not required if:

- Event source reclaims state (e.g. synchronous invocations)
- Event source has persistent storage (e.g. S3)

OPTION #1 - STREAMS

Benefits

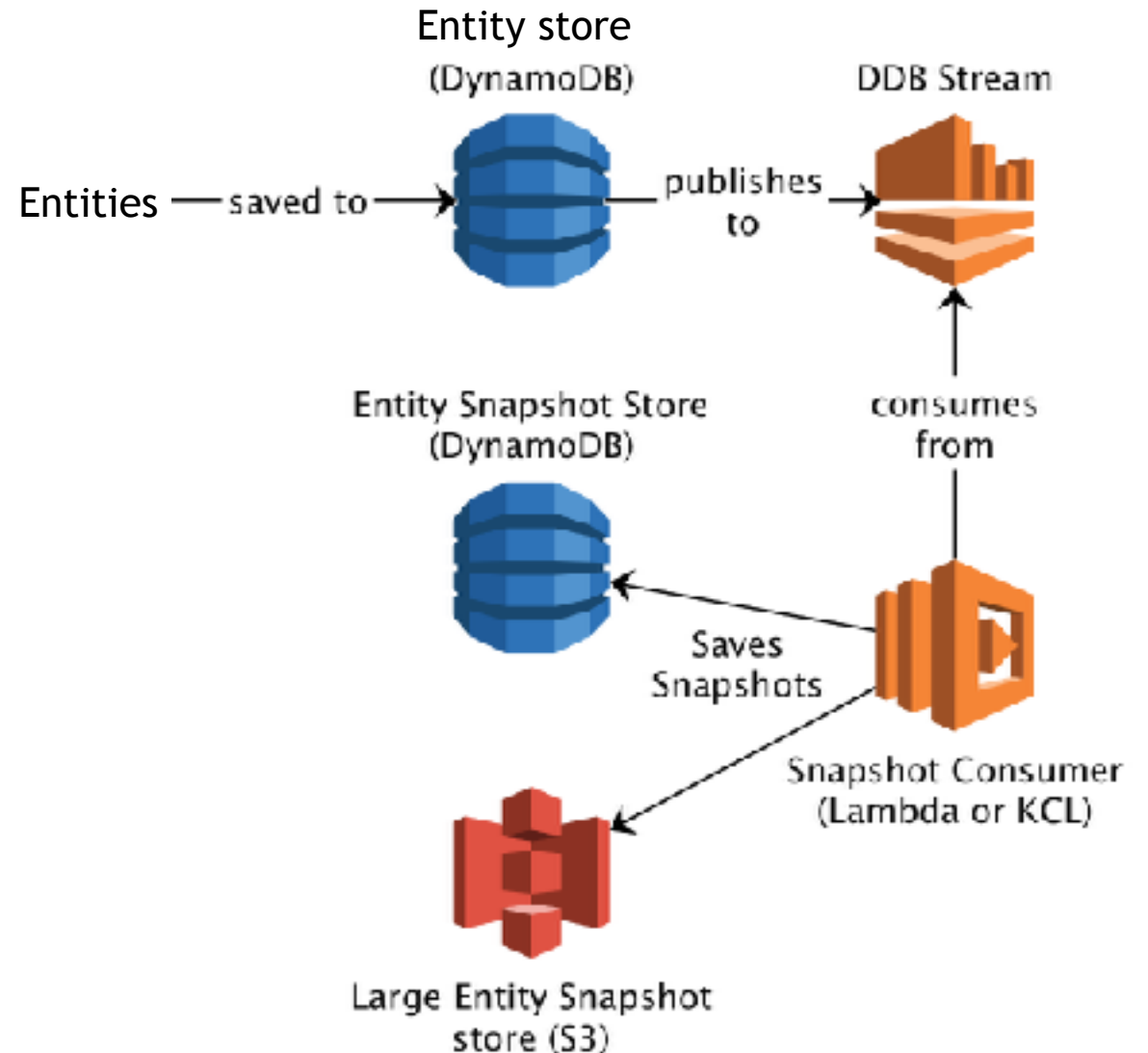
- Ordered processing on events
- Multiple consumers processing same event list

• Downsides

- Need to manage sharding and scaling
- Restricted concurrency
- Complex routing/filtering rules

• Scenarios

- Event sourcing (e.g. replication)
- Aggregations



OPTION #2 - QUEUES

Benefits

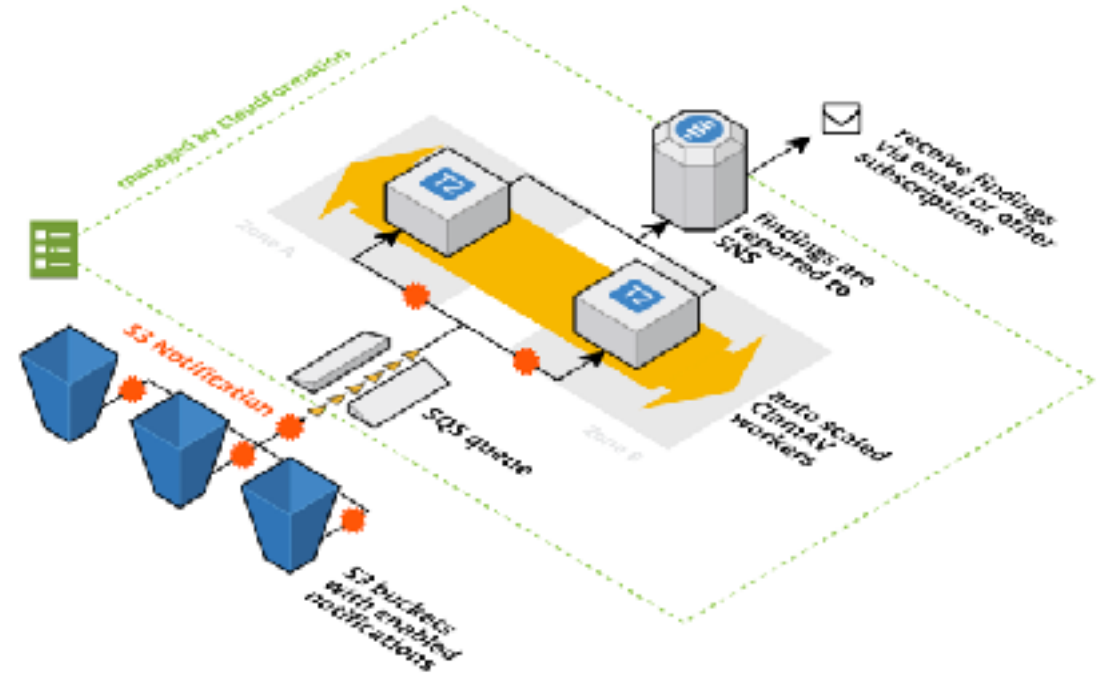
- Concurrent processing on events
- Better scaling/ease of use

• Downsides

- Limited consumers
- Order not guaranteed

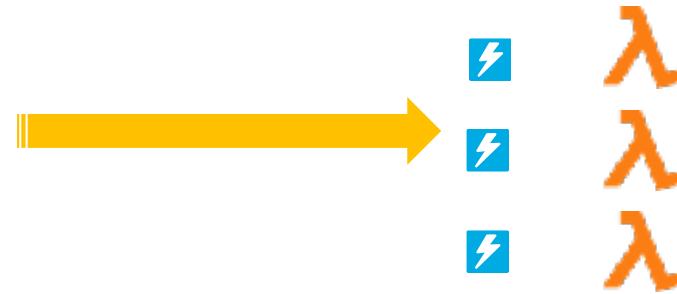
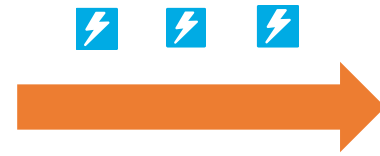
• Scenarios

- Idempotent inspections
- Parallel processing (e.g. MapReduce)

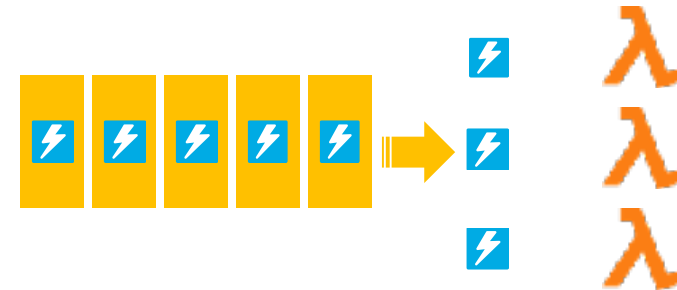


IMPACT ON SCALING/RESILIENCE

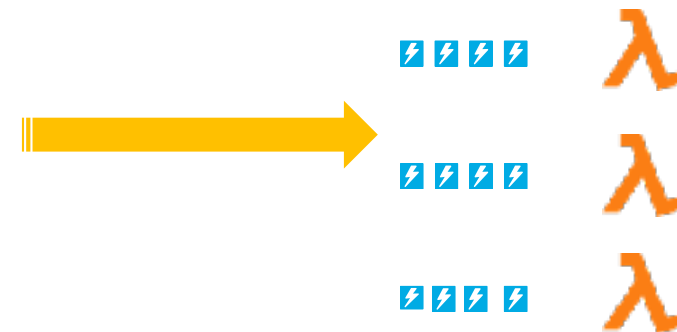
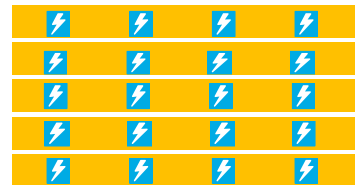
Notifications
No event store



State transfer
Queue based











State transfer
Stream based



DECISION #3 – ROUTERS

RECAP: EVENT ROUTERS

	Event source	Event router
<i>Controls...</i>	<i>Payload Retention</i>	<i>On-success semantics Filtering On-Failure semantics</i>
AWS Examples	 Amazon DynamoDB  Amazon S3  CloudWatch events (CRON)  EC2	 Amazon SNS  CloudWatch events Rules  Lambda (Event Source Mappings)  AWS IoT (Gateway)

EVENT ROUTER CHARACTERISTICS

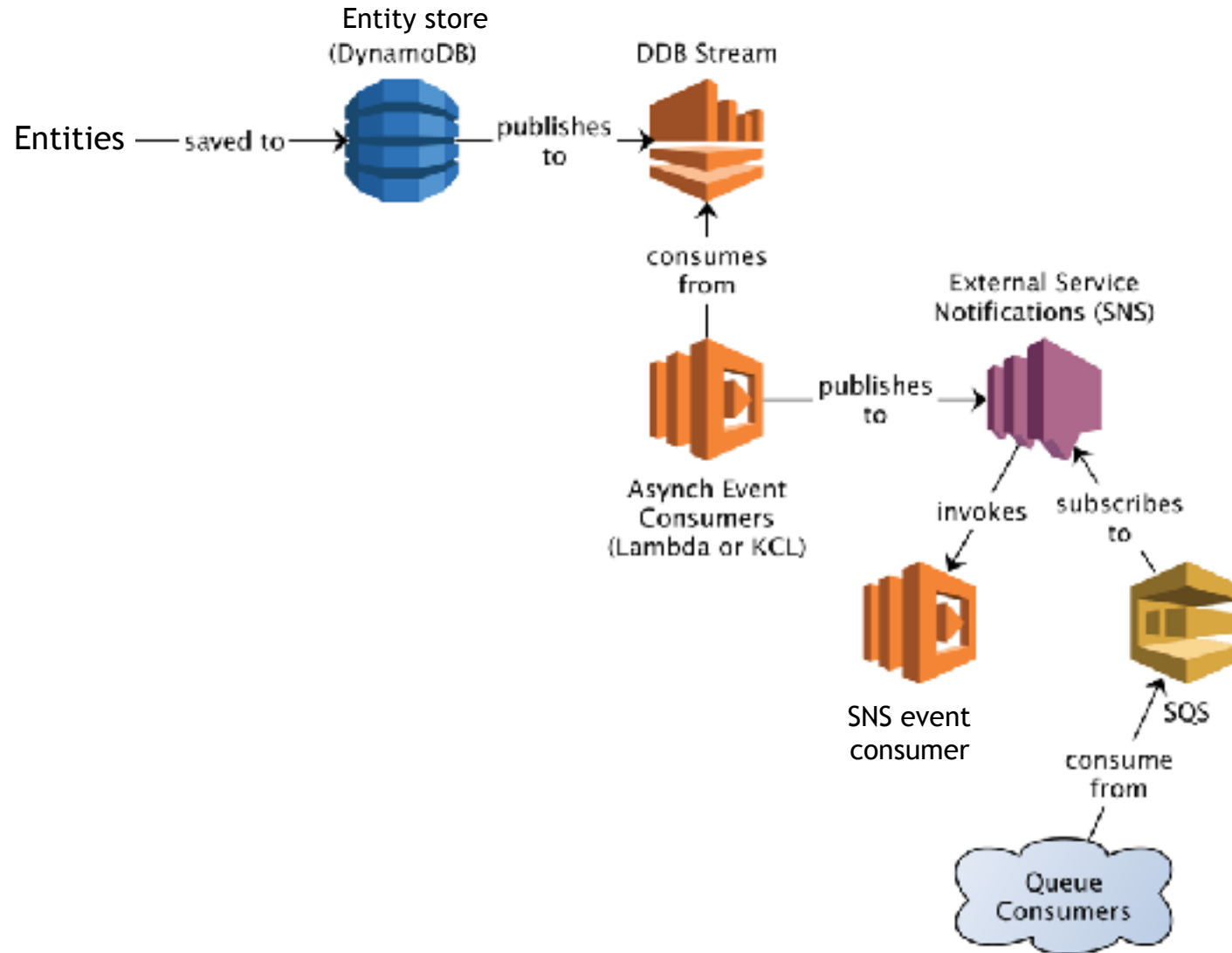
Must have

- **Pub/sub:** Securely map arbitrary sources/stores and destinations
- **Conditionals:** Ability to discard uninteresting events
 - E.g. S3 prefix filters
- **On-failure hooks:** Specific behavior if events are unprocessed
 - E.g. Lambda retries and Dead Letter Queues

Ideal

- **Bidirectional discovery:** Discovery/registry for all available sources/stores and destinations
- **Combinations:** allows joins merges between multiple event sources
- **Multiplexing:** many-to-many combinations of sources and destinations

BRINGING IT ALL TOGETHER



RECAP

1. Be smart about what goes into your payload.
2. Surface an event store when appropriate.
3. Reuse standard routers where possible.



**Supercharge the
event driven and
serverless journey
for you and your
customers.**

Thank you!