Serverless Architecture for Powerful Data Pipelines

Jason A Myers

juiceanalytics

Credit: Horst Felske and Fritz Schiemann
ex-convex.org

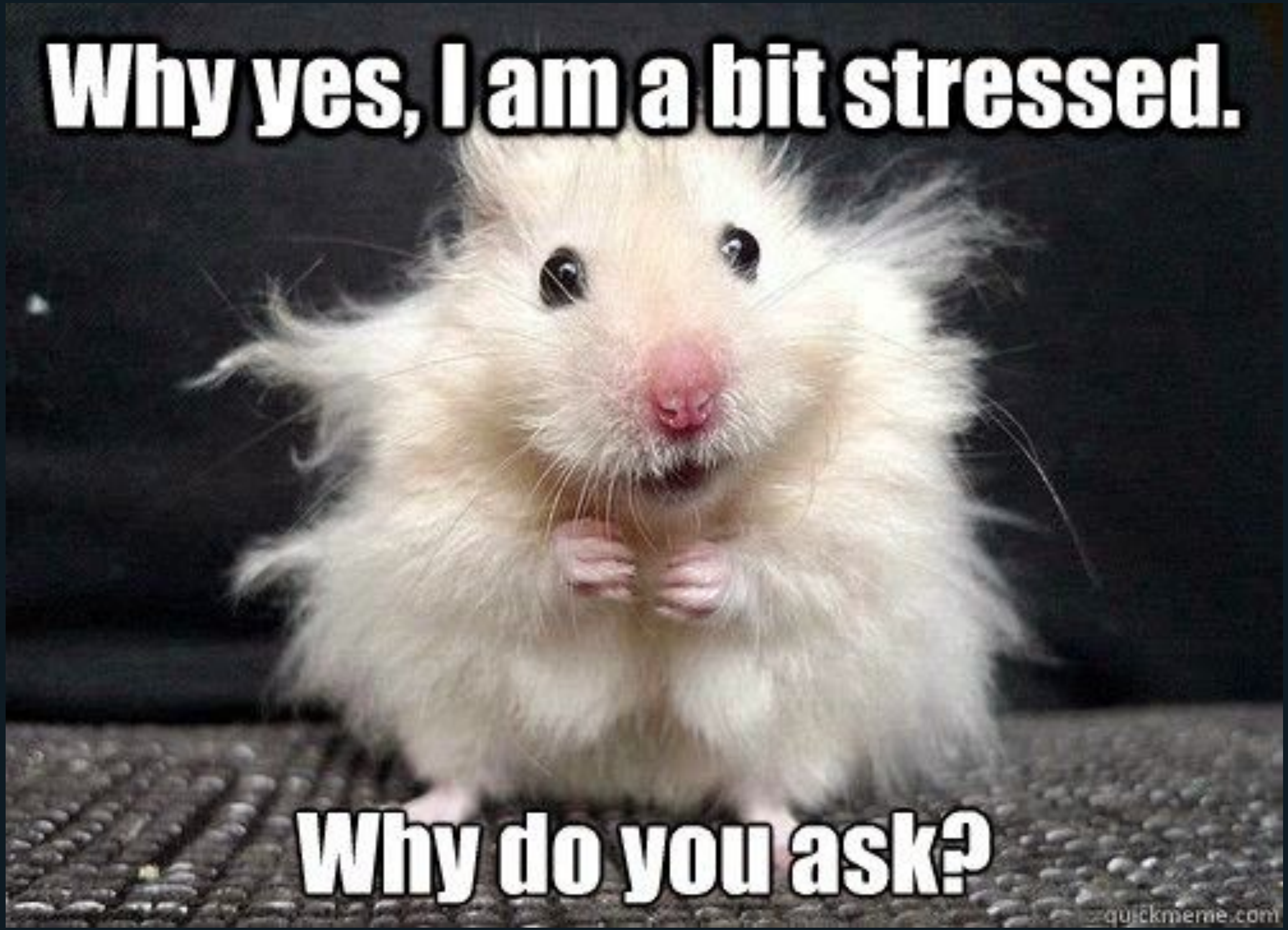
Issues

— Scheduled

— Complex

— Scaling

— Recovery



Credit: Anonymous

Python Toolkits and Services

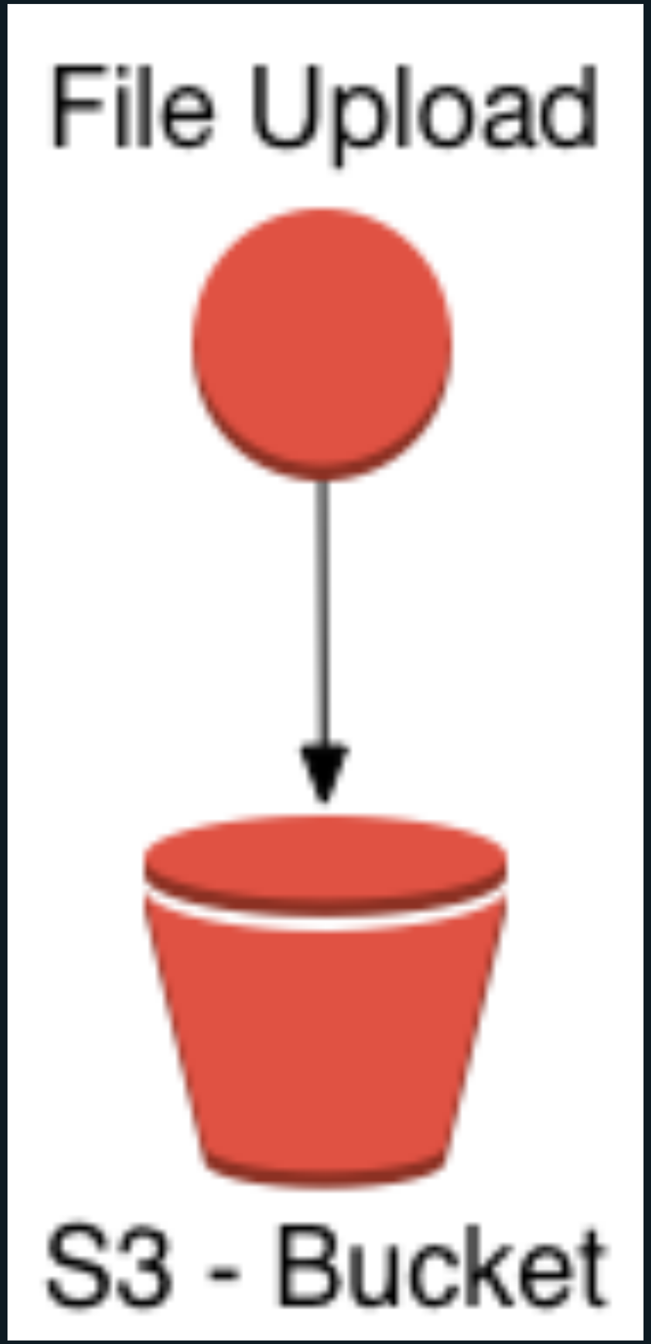
- Luigi
- Airflow
- AWS Data Pipeline

SERVERLESS

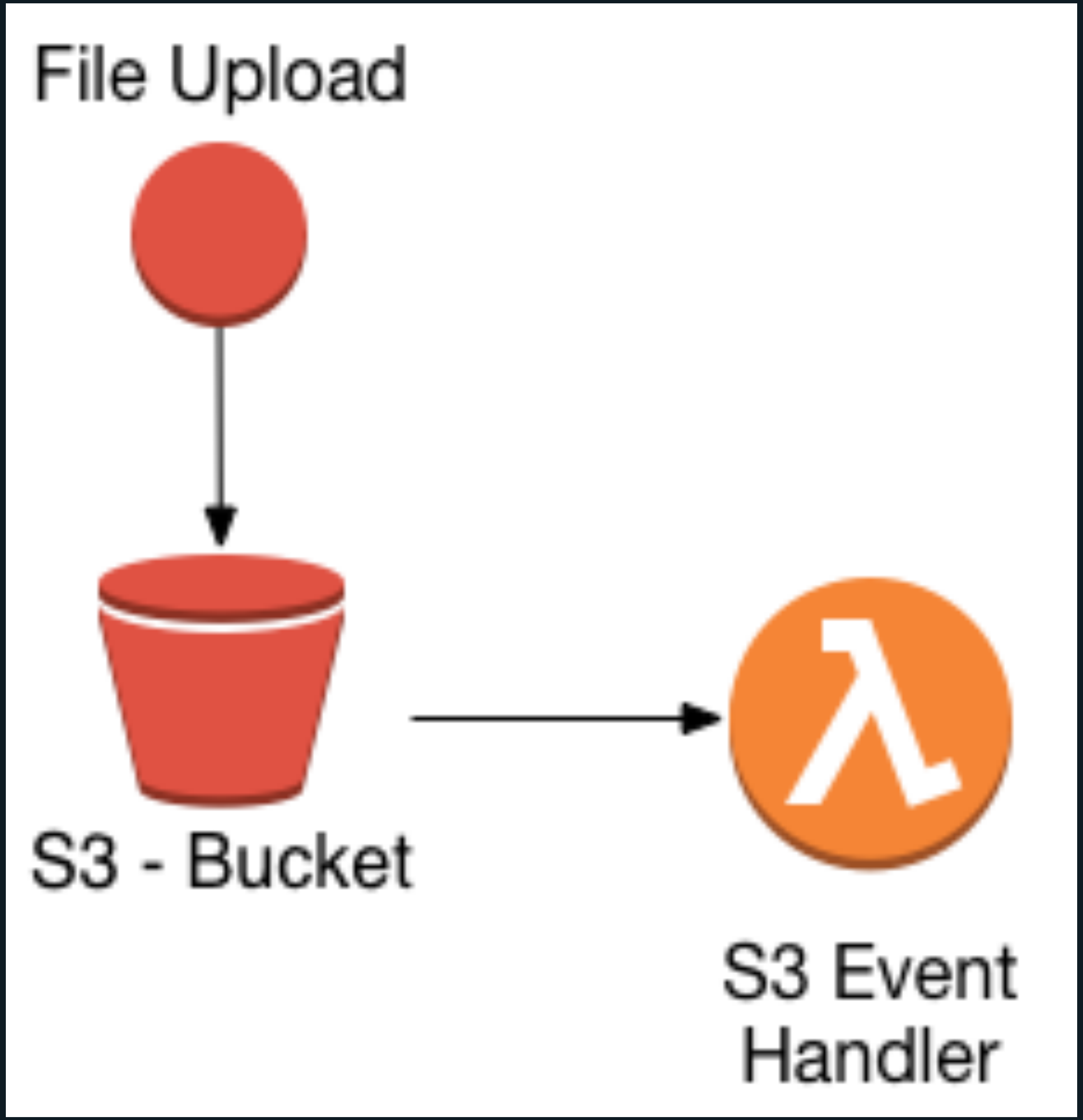
MUCH HYPE!
VERY EXCITE
memegenerator.net

Cloud Functions

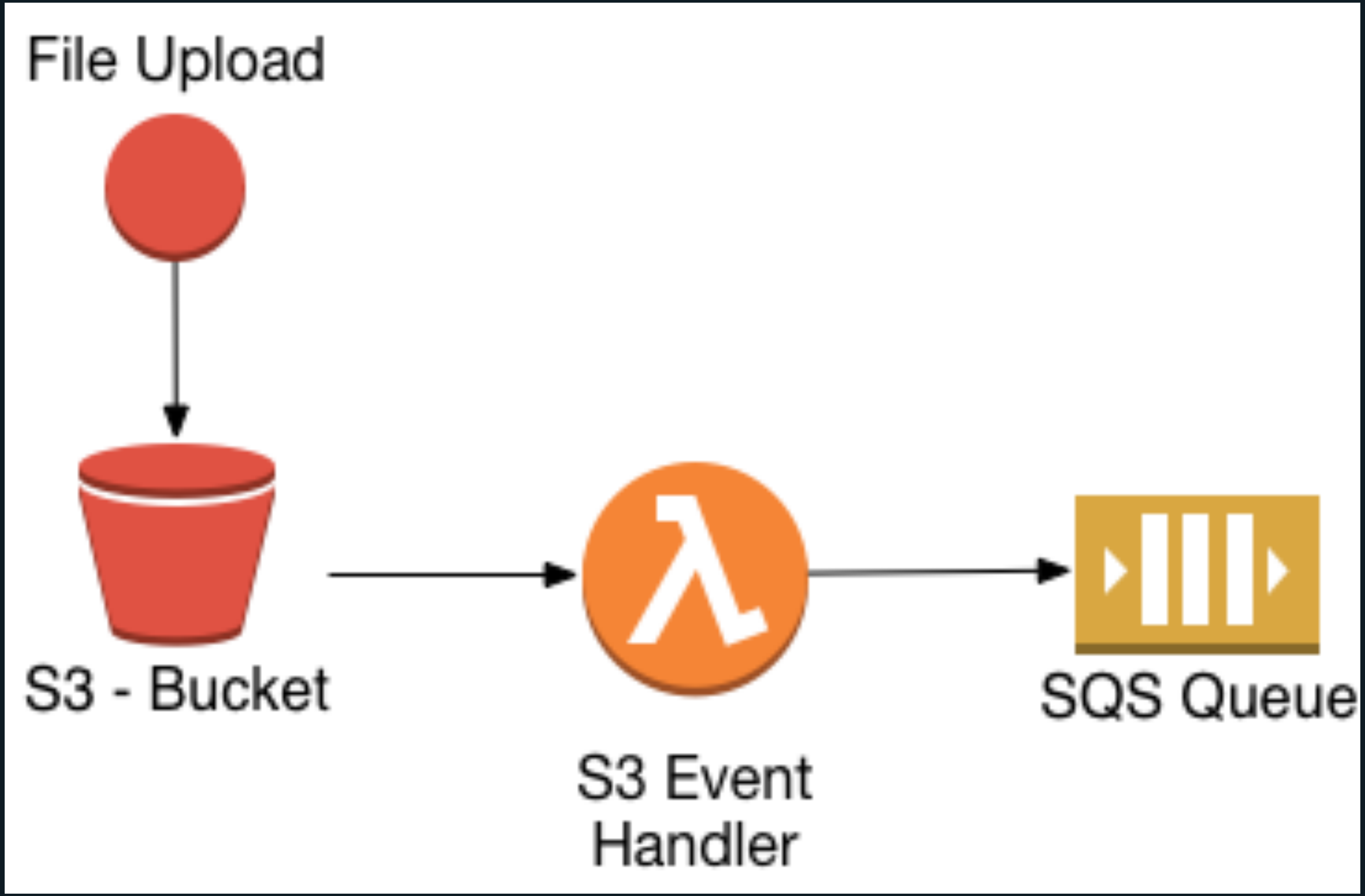
Simple Pipeline overview



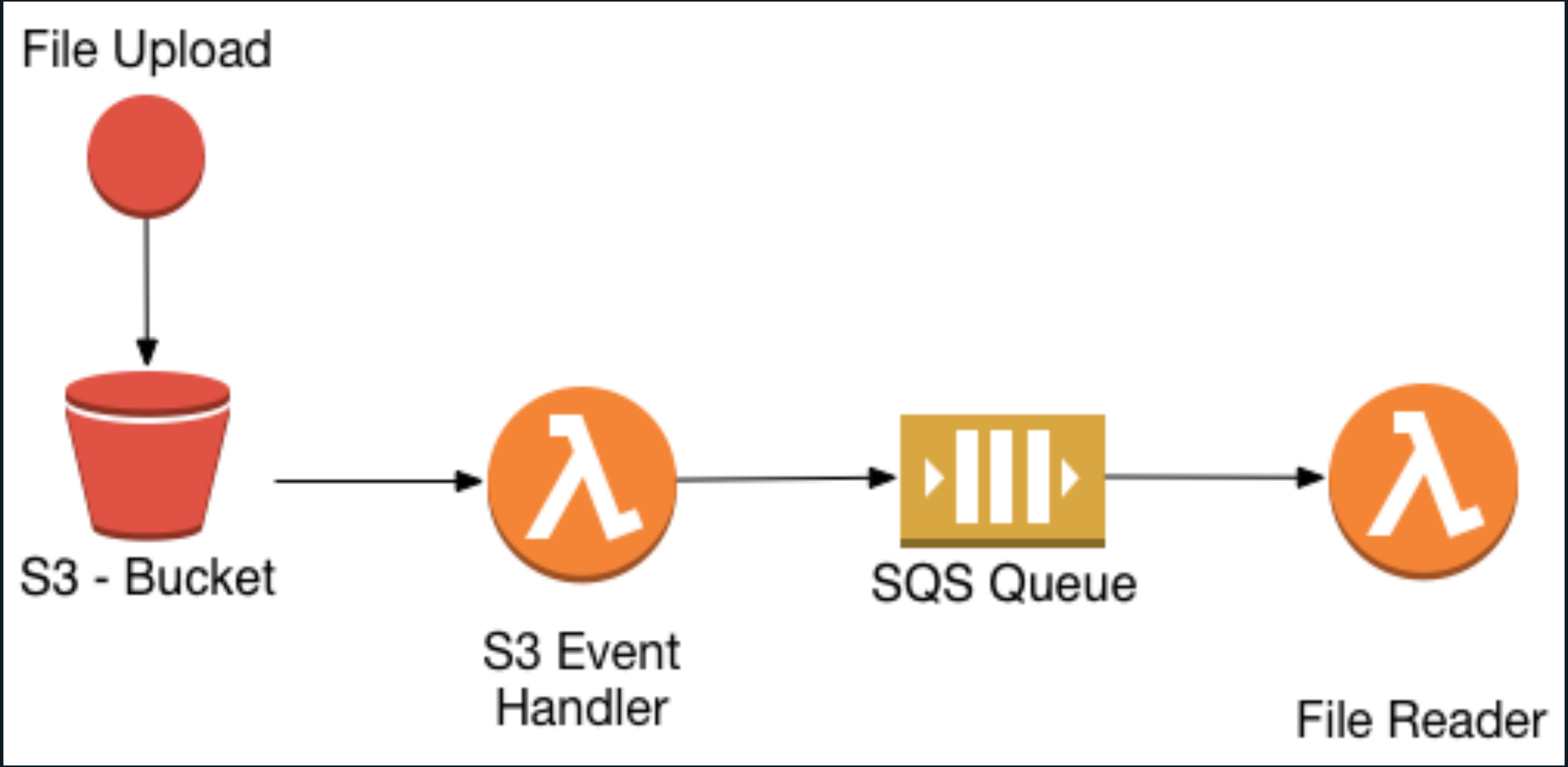
Simple Pipeline overview



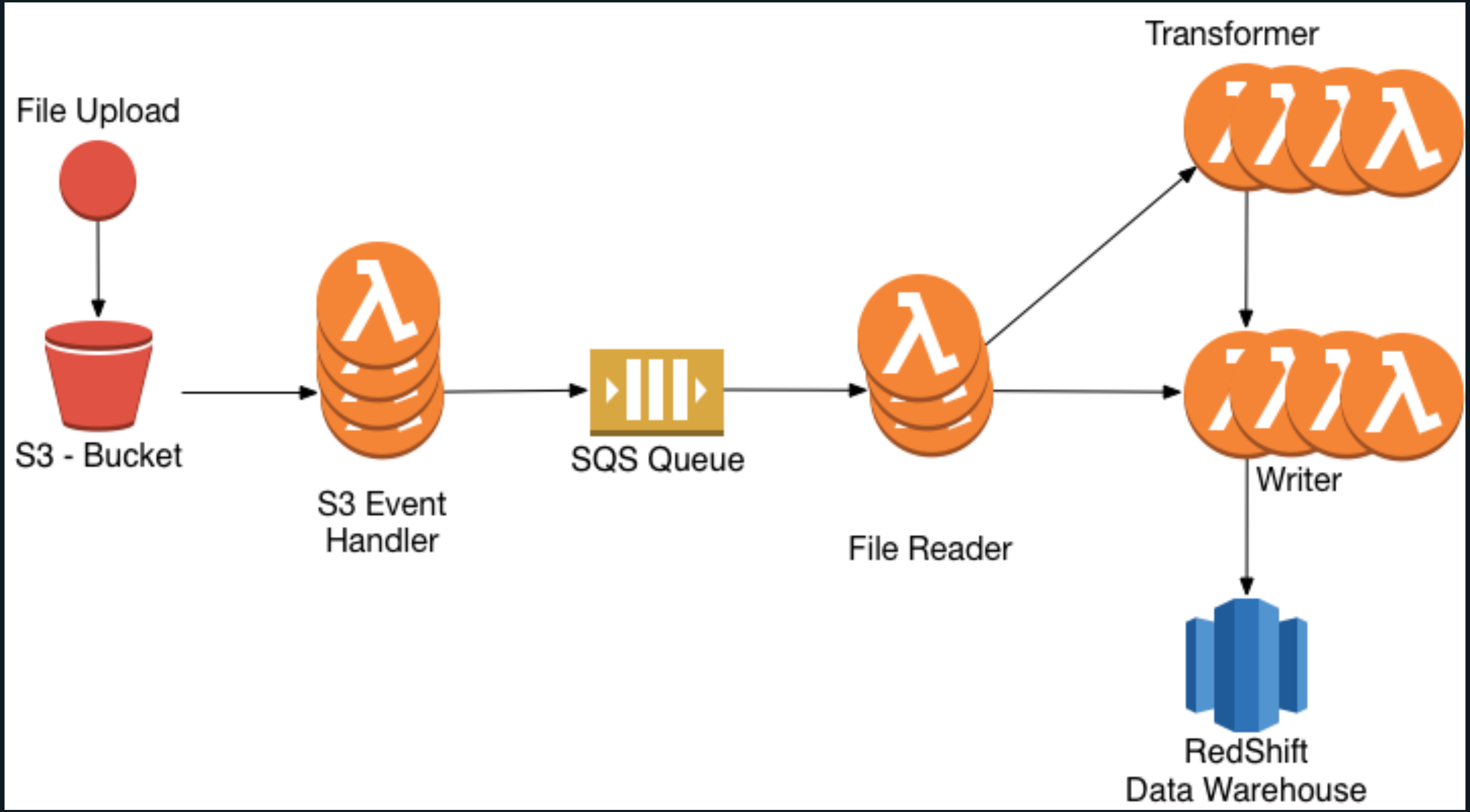
Simple Pipeline overview



Simple Pipeline overview



Simple Pipeline overview



Serverless Python Tools

- Zappa
- Apex
- Chalice
- Serverless

Apex

- Multiple Environment Support
- Function Deployment
- Infrastructure as code via Terraform

Project Structure

```
├── functions
|   └── listener
|       └── main.py
├── infrastructure
|   ├── dev
|   |   ├── main.tf
|   |   ├── outputs.tf
|   |   └── variables.tf
|   ├── prod
|   |   ├── main.tf
|   |   ├── outputs.tf
|   |   └── variables.tf
├── project.json
└── project.prod.json
```

Project Structure

```
├── functions
|   └── listener
|       └── main.py
├── infrastructure
|   ├── dev
|   |   ├── main.tf
|   |   ├── outputs.tf
|   |   └── variables.tf
|   ├── prod
|   |   ├── main.tf
|   |   ├── outputs.tf
|   |   └── variables.tf
├── project.json
└── project.prod.json
```

Project Structure

```
├── functions
|   └── listener
|       └── main.py
├── infrastructure
|   ├── dev
|   |   ├── main.tf
|   |   ├── outputs.tf
|   |   └── variables.tf
|   ├── prod
|   |   ├── main.tf
|   |   ├── outputs.tf
|   |   └── variables.tf
├── project.json
└── project.prod.json
```

Apex package.json

```
{
  "name": "listener",
  "description": "S3 File Listener",
  "runtime": "python3.6",
  "memory": 128,
  "timeout": 5,
  "role": "arn:aws:iam::ACOUNTNUM:role/listen_lambda_function",
  "environment": {},
  "defaultEnvironment": "dev"
}
```

S3 Event Handler

```
import logging

import boto3

log = logging.getLogger()
log.setLevel(logging.DEBUG)

def get_bucket_key(event);
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = event['Records'][0]['s3']['object']['key']
    return bucket, key

def handle(event, context):
    log.info('{}-{}'.format(event, context))
    bucket_name, key_name = get_bucket_key(event)
```

S3 Event Handler

```
import logging

import boto3

log = logging.getLogger()
log.setLevel(logging.DEBUG)

def get_bucket_key(event);
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = event['Records'][0]['s3']['object']['key']
    return bucket, key

def handle(event, context):
    log.info('{}-{}'.format(event, context))
    bucket_name, key_name = get_bucket_key(event)
```

S3 Event Handler

```python
import logging

import boto3

log = logging.getLogger()
log.setLevel(logging.DEBUG)

def get_bucket_key(event);
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = event['Records'][0]['s3']['object']['key']
    return bucket, key

def handle(event, context):
    log.info('{}-{}'.format(event, context))
    bucket_name, key_name = get_bucket_key(event)
```

S3 Event Handler (cont.)

```
values = {
    'bucket_name': bucket_name,
    'key_name': key_name,
    'timestamp': datetime.utcnow().isoformat()
}

client = boto3.client('sqs')
client.publish(
    TopicArn=topic_arn,
    Message=json.dumps(values)
)
```

AWS Logging Permissions

```
data "aws_iam_policy_document" "listener_logging" {
  statement {
    sid       = "AllowRoleToOutputCloudWatchLogs"
    effect    = "Allow"
    actions   = ["logs:*"]
    resources = ["*"]
  }
}

resource "aws_iam_policy" "listener_logs" {
  name        = "listener_logs"
  description = "Allow listener to log operations"
  policy      = "${data.aws_iam_policy_document.listener_logging.json}"
}
```

AWS IAM Role Assumption

```
data "aws_iam_policy_document" "listener_lambda_assume_role" {
  statement {
    sid     = "AllowRoleToBeUsedbyLambda"
    effect  = "Allow"
    actions = ["sts:AssumeRole"]

    principals {
      type        = "Service"
      identifiers = ["lambda.amazonaws.com"]
    }
  }
}

resource "aws_iam_role" "listener_lambda_function" {
  name               = "listener_lambda_function"
  assume_role_policy = "${data.aws_iam_policy_document.listener_lambda_assume_role.json}"
}
```

AWS Policy Attachment

```
resource "aws_iam_policy_attachment" "listener_logs_attach" {
  name       = "listener_logs_attach"
  roles      = ["${aws_iam_role.listener_lambda_function.name}"]
  policy_arn = "${aws_iam_policy.listener_logs.arn}"
}
```

Deploy Function and Infrastructure

```
# apex deploy
# apex infra plan
# apex infra apply
```

What about those other functions…

Lambda Packages

Zappa Project/Gun.io

Lambda	Packages	List
bcrypt	cffi	PyNaCl
datrie	LXML	misaka
MySQL-Python	numpy	OpenCV
Pillow (PIL)	psycopg2	PyCrypto
cryptography	pyproj	python-ldap
python-Levenshtein	regex	

Dependency Handling in Apex

```
"hooks":{
    "build": "pip install -r requirements.txt -t ."
}
```

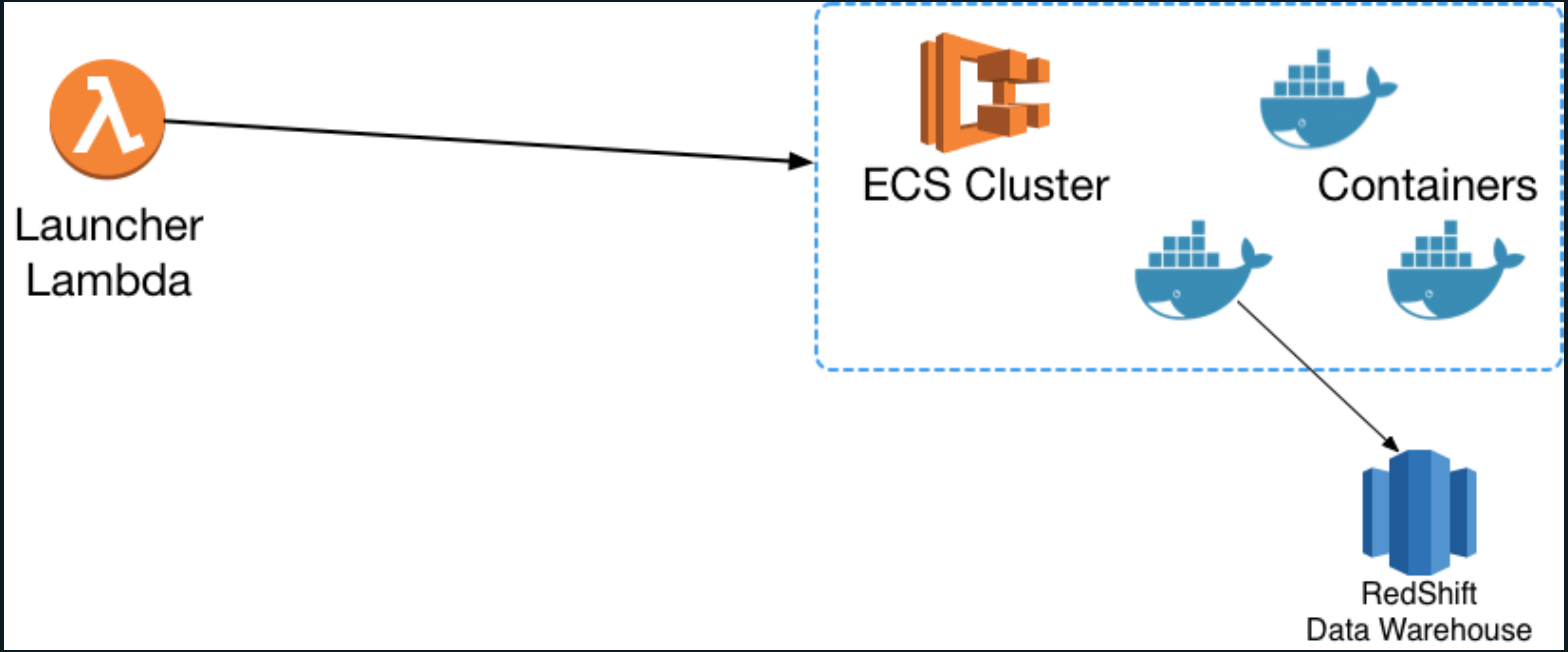
Function Considerations

- atomic
- idempotent

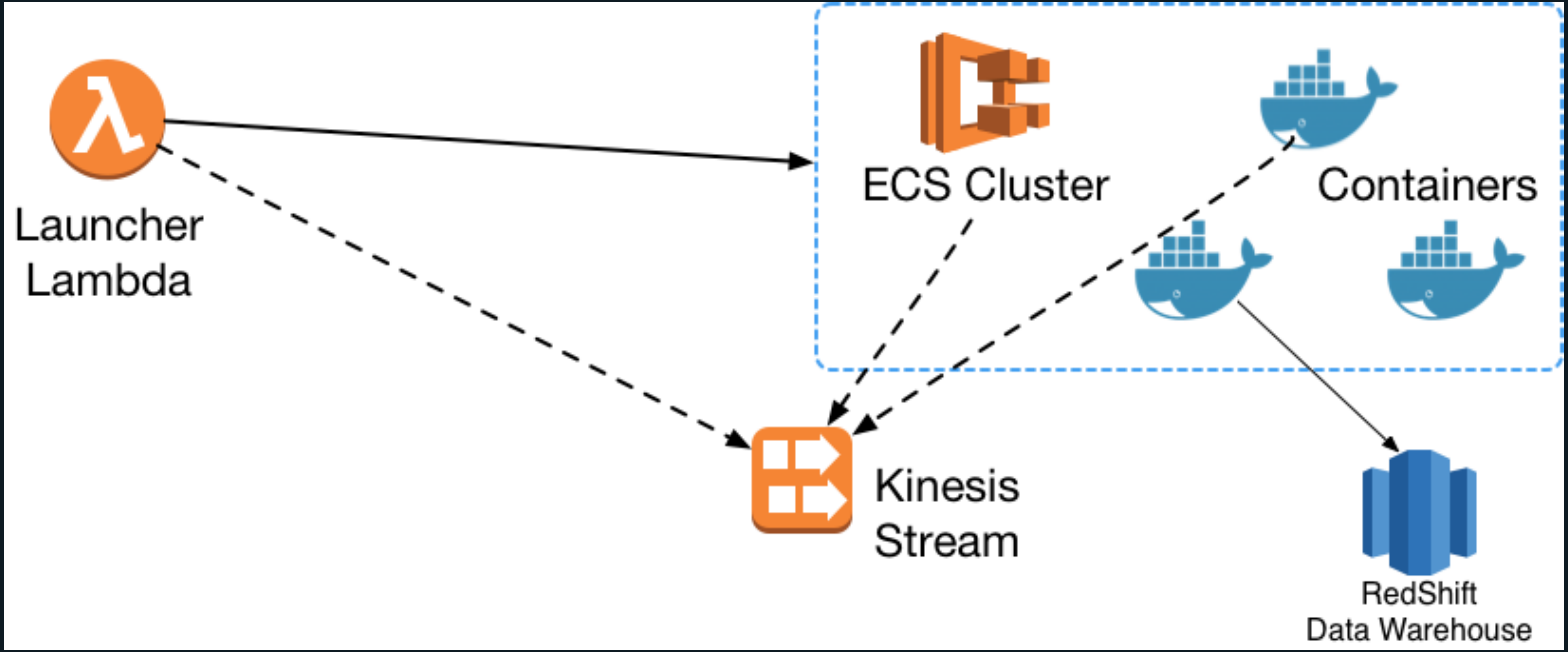
Dis is the Remix

- Longer Jobs
- Legacy Pipelines

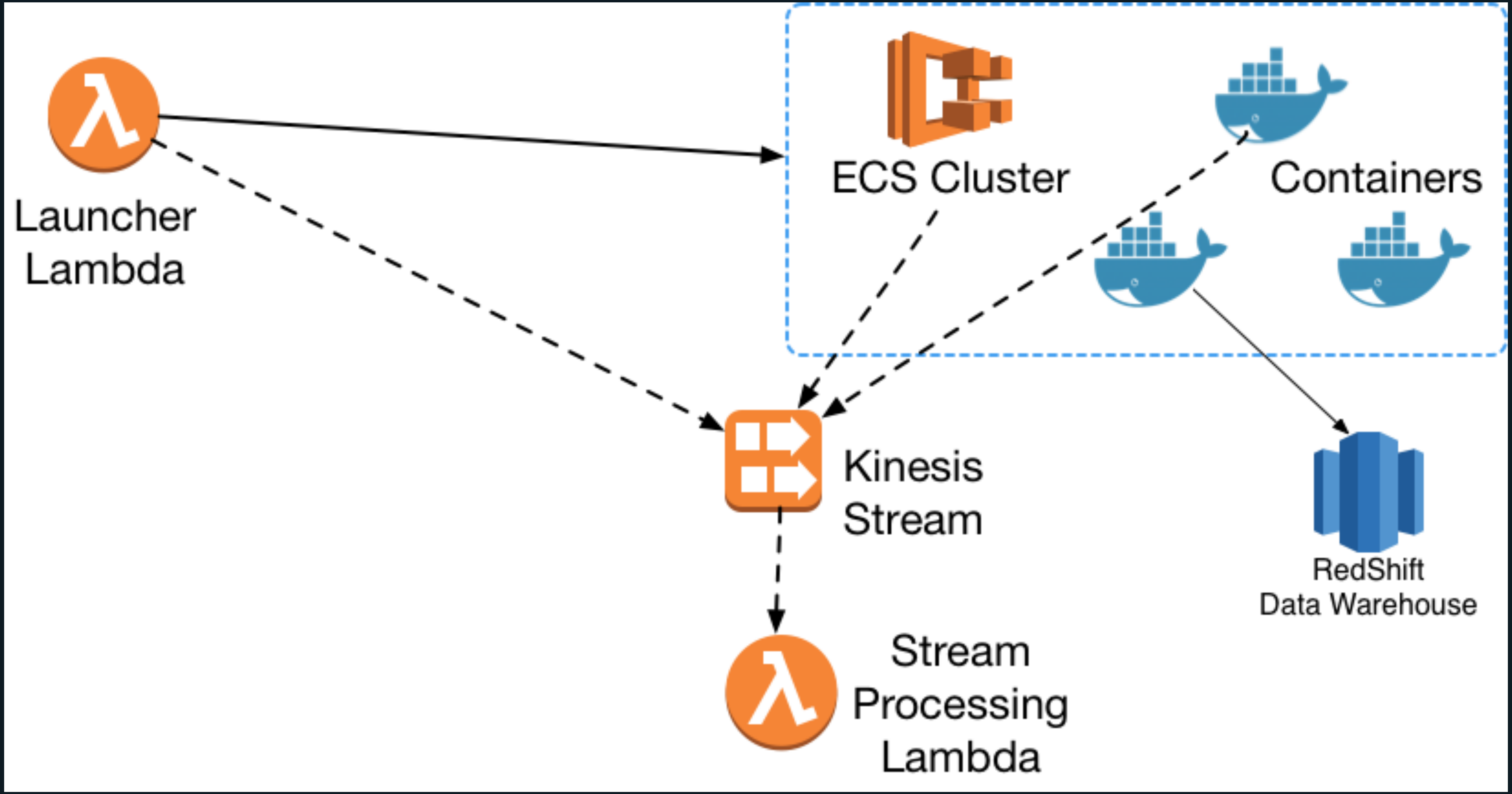
Hybrid Pipeline overview



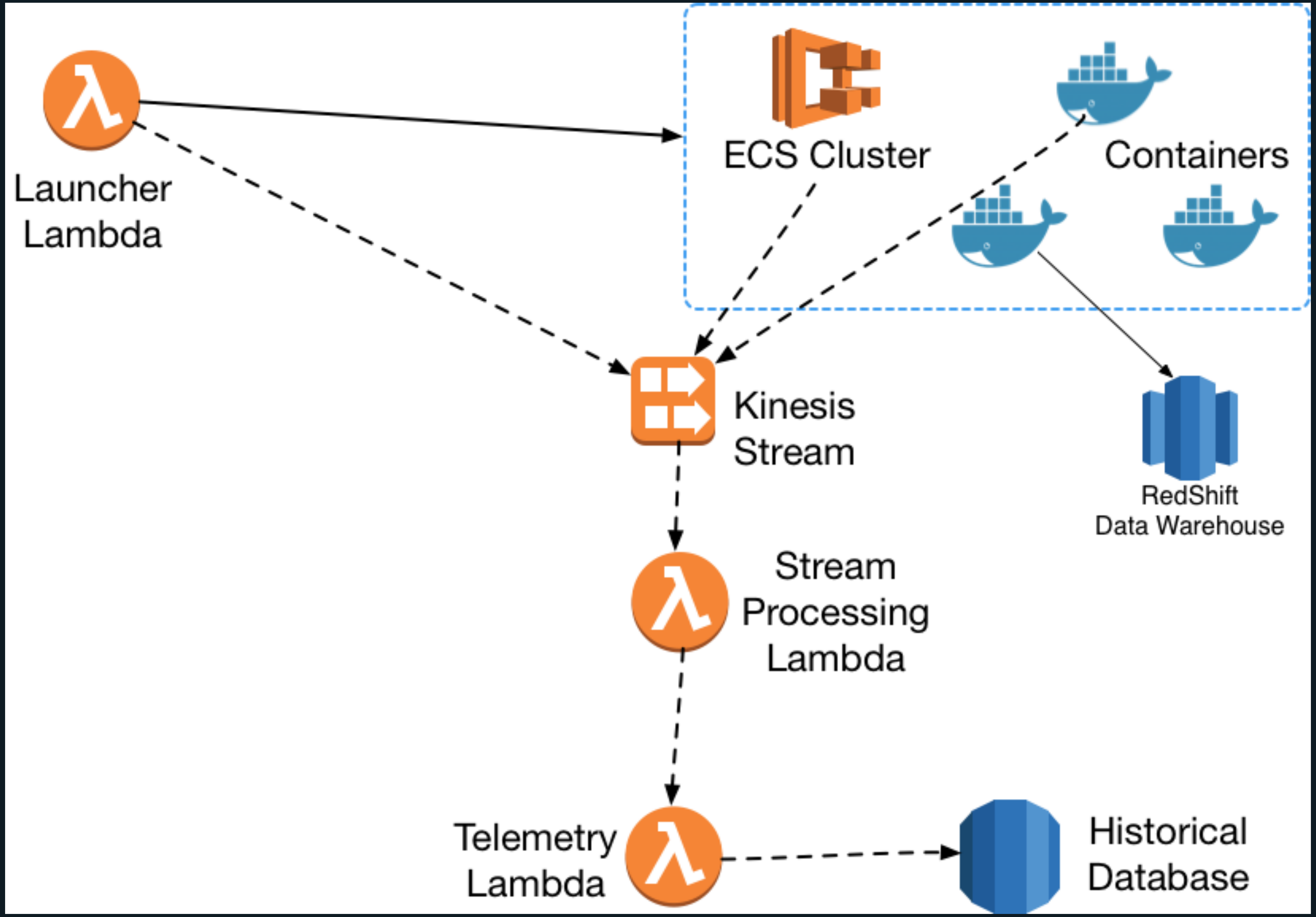
Hybrid Pipeline overview



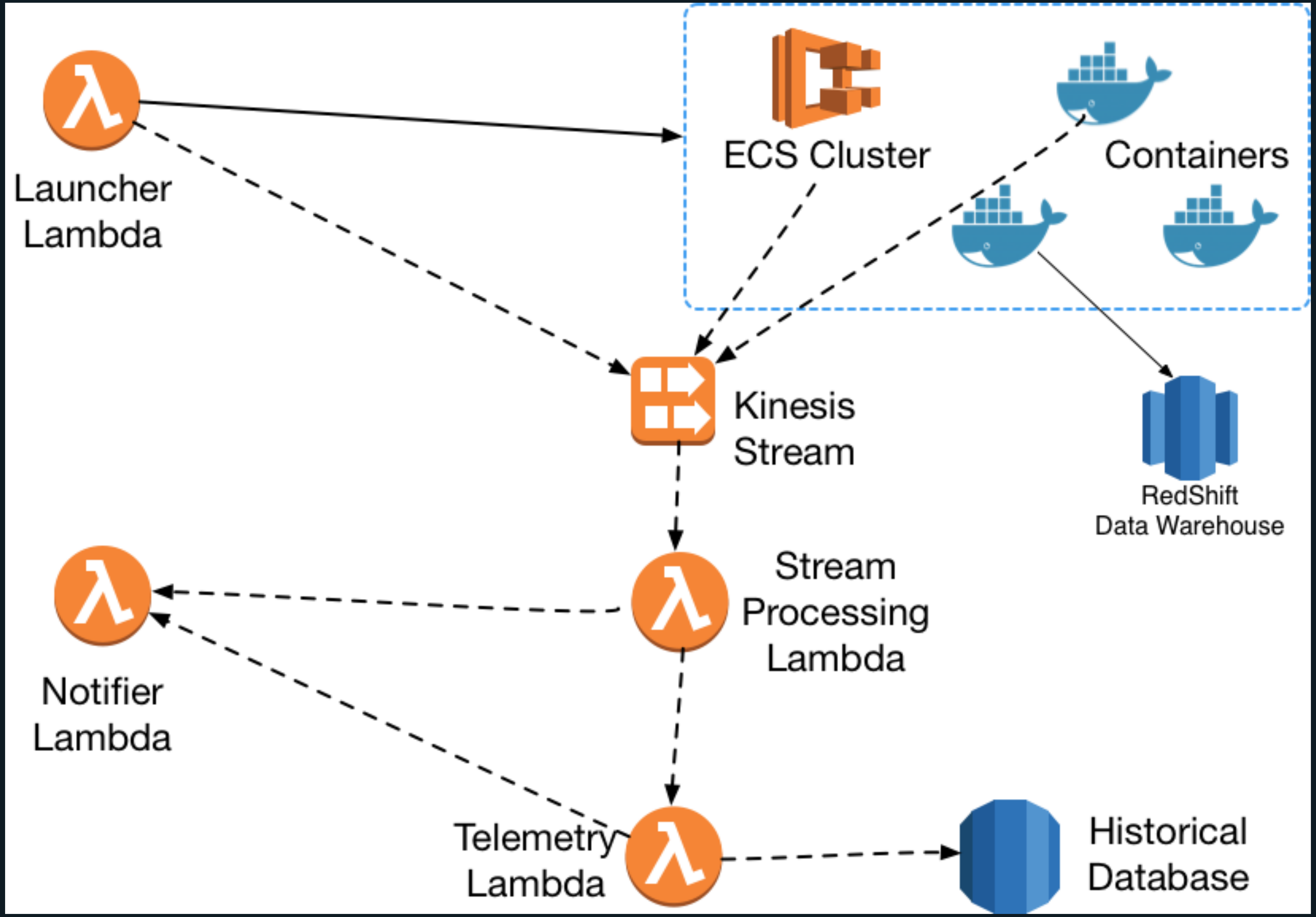
Hybrid Pipeline overview



Hybrid Pipeline overview



Hybrid Pipeline overview



Closing thoughts

- Workload and Phases are important
- 14x improvement
- 0.73x improvement

Questions?