
SKYWATCH
Delivering Intelligence from space


Crop Forecasting


Scanning...
Pipeline Monitoring


Plane & Ship Tracking


Market Intelligence Gathering

Distributed Data Engineering - Lessons

Distributed Data Engineering - Lessons

1. Metrics

Distributed Data Engineering - Lessons

1. Metrics
2. Logging

Distributed Data Engineering - Lessons

1. Metrics
2. Logging
3. Frameworks

Distributed Data Engineering - Lessons

1. Metrics
2. Logging
3. Frameworks
4. Serverless ETL

1. Metrics were lacking

Before

user_id	num_downloads	num_uploads
4	10	1
7	6	3

Before

user_id	num_downloads	num_uploads
4	11 +1	1
7	6	3

downloads

date	user_id	image_size
2017-10-01 14:40:32	4	1365
2017-10-02 11:01:11	4	650

downloads

date	user_id	image_size
2017-10-01 14:40:32	4	1365
2017-10-02 11:01:11	4	650
2017-10-02 11:06:00	5	9001

Source of Truth

Database

Source of Truth

Database

- Migration headaches
- Manage connections
- Performance

Source of Truth

~~Database~~

Logging

```
{
    "message": "Downloaded img",
    "userId": "1234",
    "imgId": "1d3x5",
    "service": "download-server",
 "time": "1509385330"
}
```

Two Kinds of Logs

Server logs

- **Debugging**
- **Support**

```
[Wed Oct 11 14:32:12 2000] [info] [client
127.0.0.1] image 1d3x5 downloaded by userId 1234
```

Two Kinds of Logs

Server logs

- **Debugging**
- **Support**

```
[Wed Oct 11 14:32:12 2000] [info] [client
127.0.0.1] image 1d3x5 downloaded by userId 1234
```

Metric logs

- **Dashboards**
- **Analytics**

```
{
    "message": "Downloaded img",
    "userId": "1234",
    "imgId": "1d3x5",
    "service": "download-server",
 "time": "1509385330"
}
```

Metric logs

- Dashboards
- Analytics

```
{
    "message": "Downloaded img",
    "userId": "1234",
    "imgId": "1d3x5",
    "service": "download-server",
 "time": "1509385330"
}
```

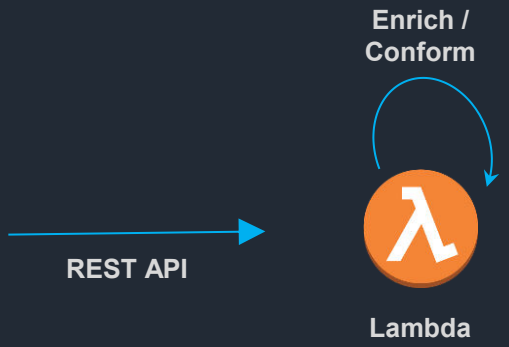
Centralize

Metric Collector

```
import observatory
obs = observatory.Tracker()

obs.track('search_made', {
    'query': event.query,
    'n_results': len(resp['data']),
    'user_id': user_item.id
})
```

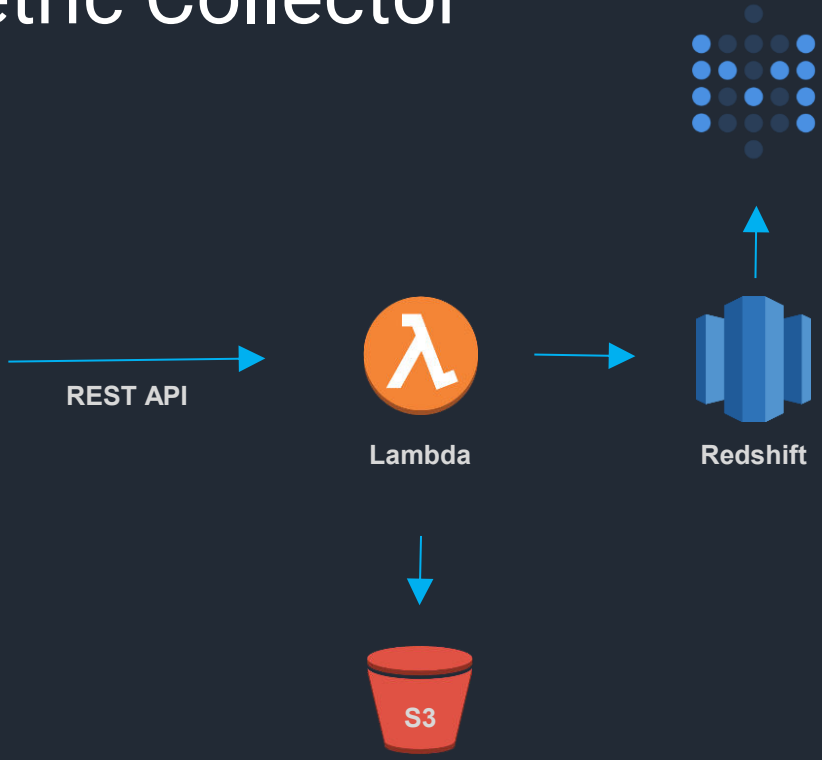
Metric Collector

```
import observatory
obs = observatory.Tracker()

obs.track('search_made', {
    'query': event.query,
    'n_results': len(resp['data']),
    'user_id': user_item.id
})
```



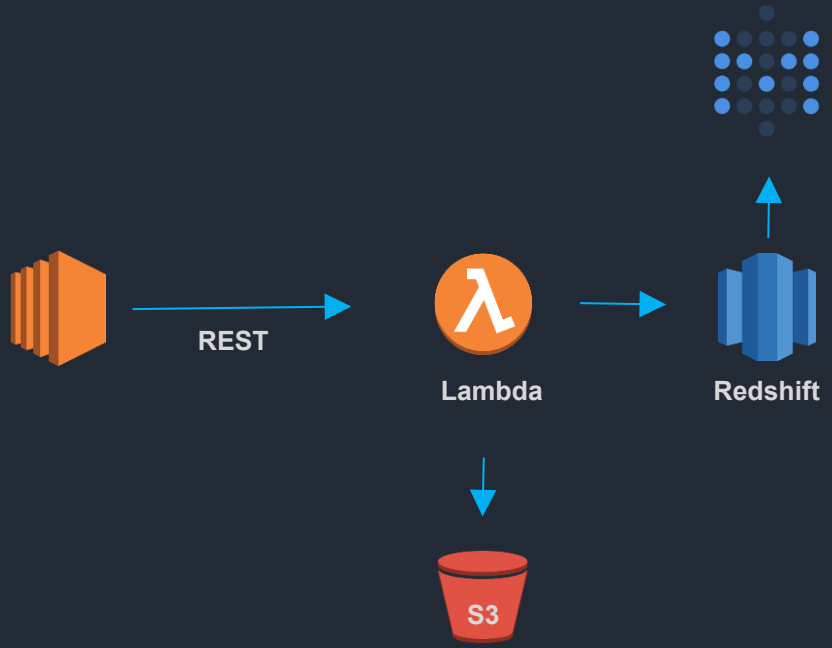
Metric Collector

```
import observatory
obs = observatory.Tracker()

obs.track('search_made', {
    'query': event.query,
    'n_results': len(resp['data']),
    'user_id': user_item.id
})
```



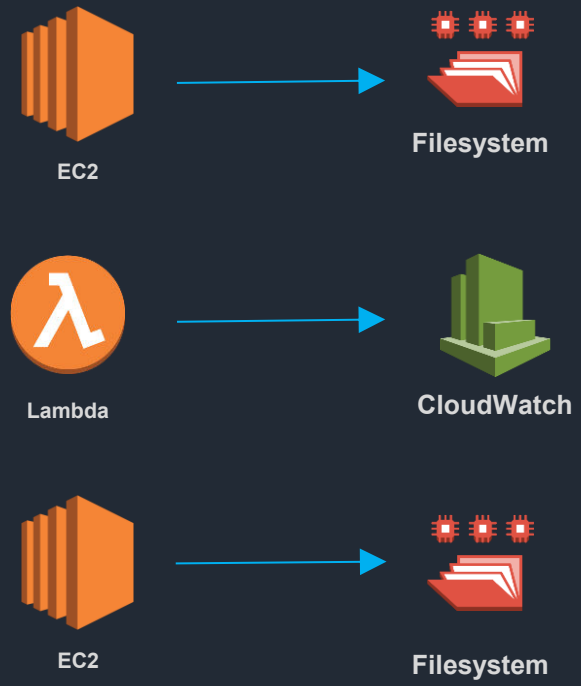
After

- Centralized metrics
- Log enrichment
- Persistent store

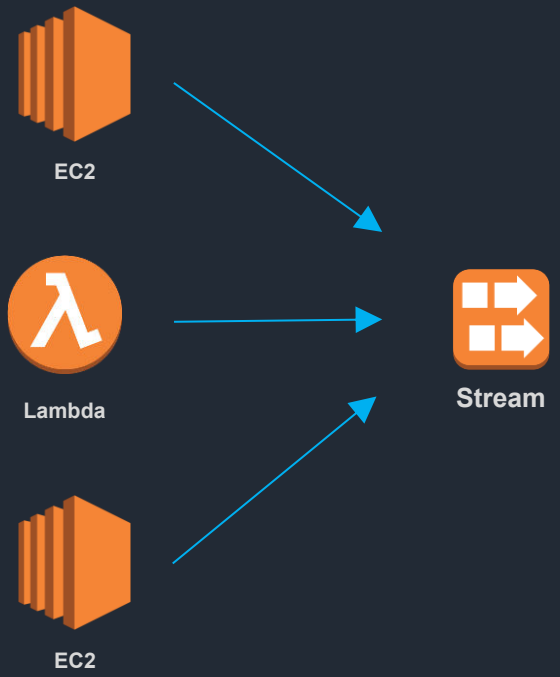


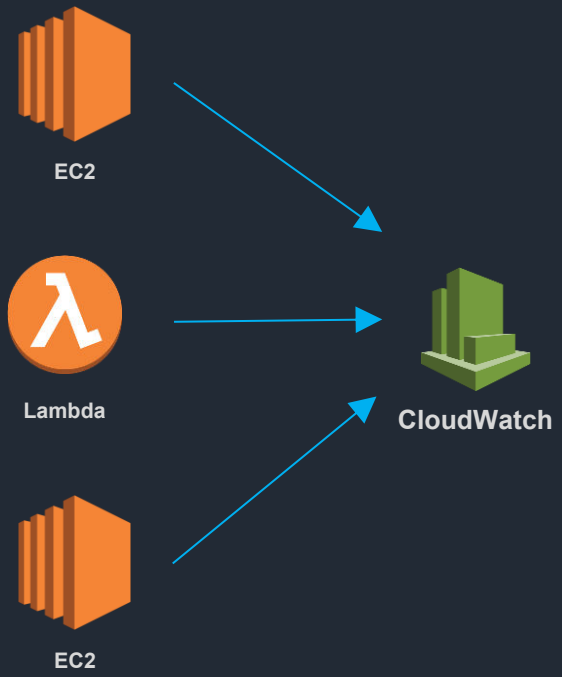
2. Debugging is painful

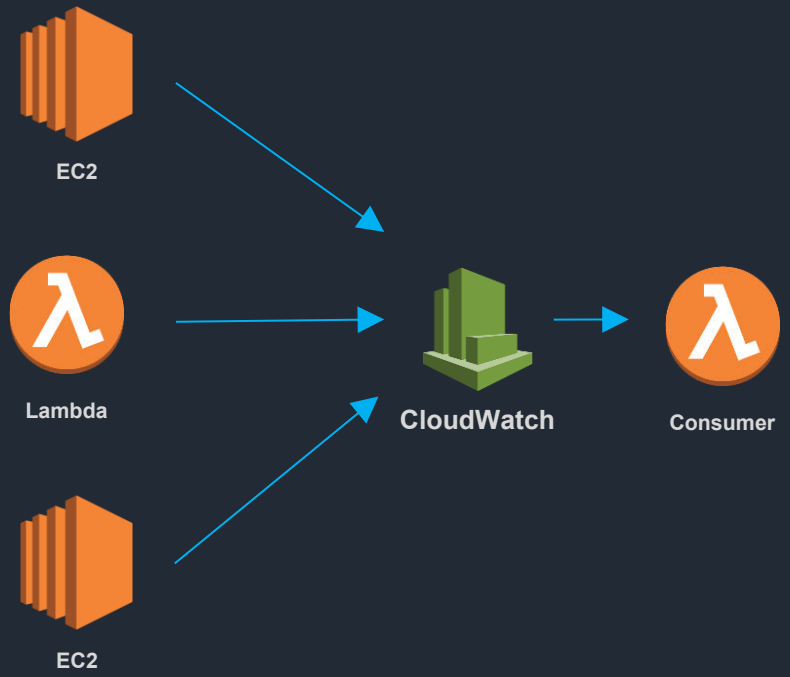
Before

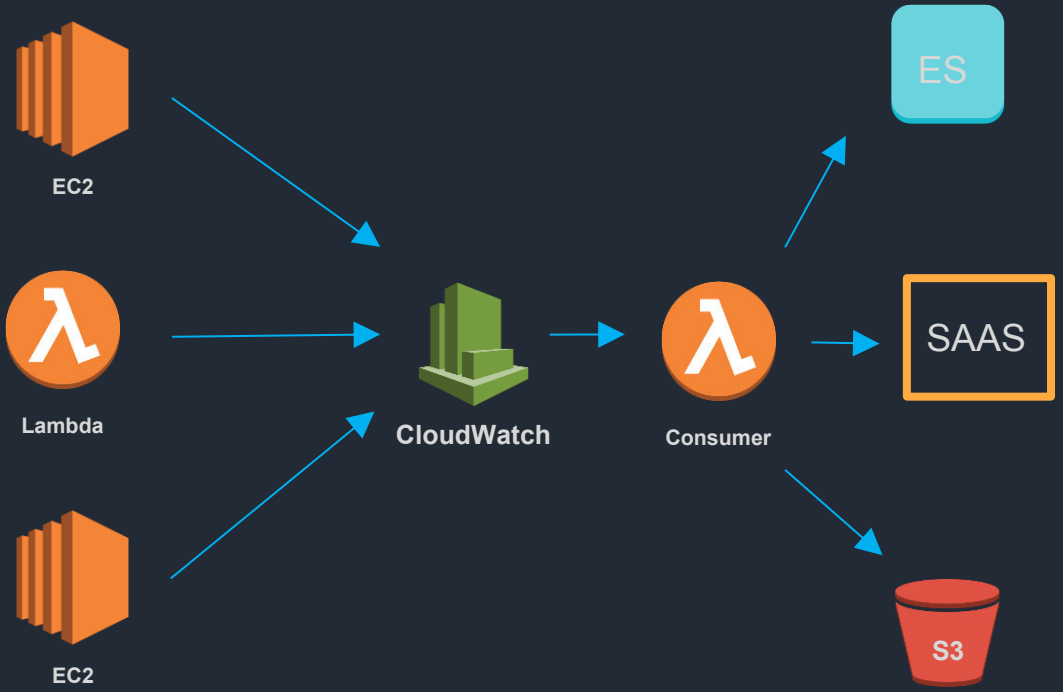


Centralize


EC2
Lambda
EC2
Stream


EC2
Lambda
EC2
CloudWatch


EC2
Lambda
EC2
CloudWatch
Consumer


EC2
Lambda
EC2
CloudWatch
Consumer
ES
SAAS
S3

After

- **Elasticsearch**
- **Search by UUID**

But!

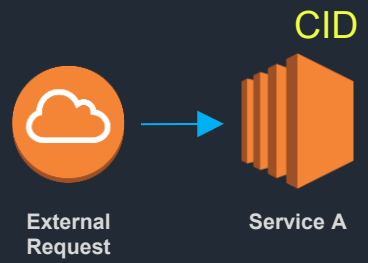
What does the full flow of a request look like?

Correlation ID

UUID

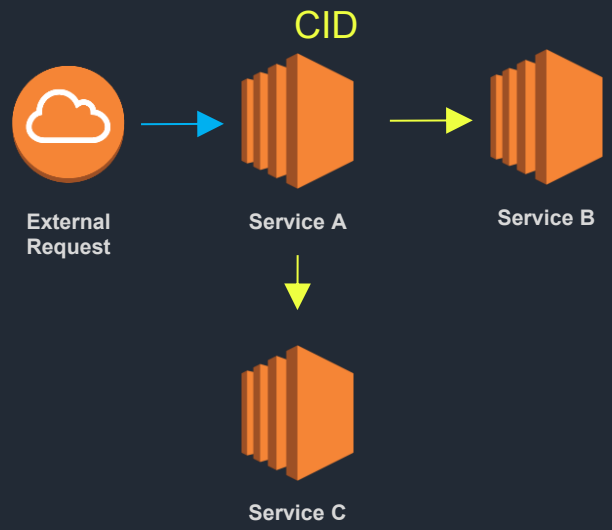
Correlation ID

- Create for any external call



Correlation ID

- CID passed everywhere



Correlation ID

In ES → filter by CID

3. Building services is slow

Before

- Online console
- Zip file deployment
- Doesn’t scale

Infrastructure as Code

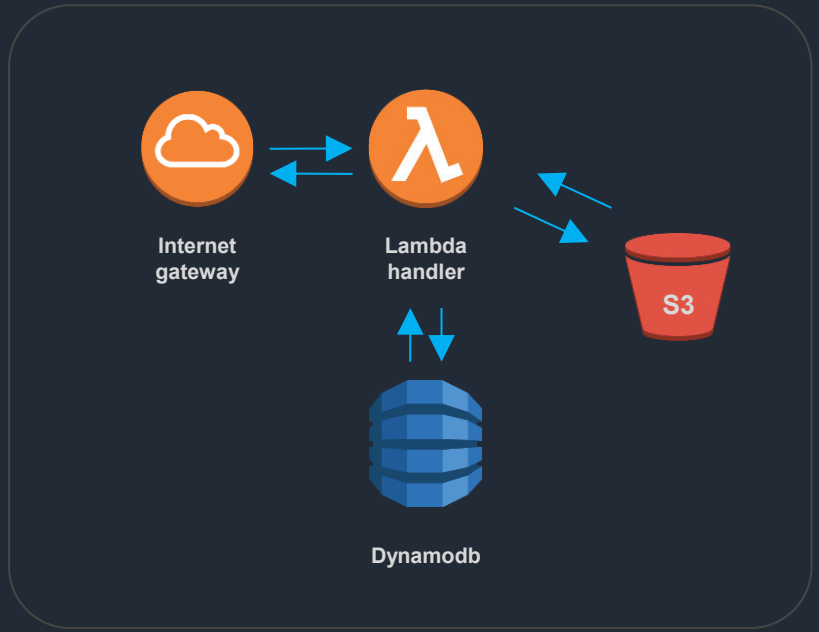
Infrastructure as Code

- Serverless Framework
- Template -> service
- Rapid deployment

```yaml
# serverless.yml

service: users

provider:
  name: aws
  runtime: nodejs6.10
  stage: dev

functions:
  usersCreate:
    handler: users.create
    events:
      - http: post users/create

resources:
  Resources:
    usersTable:
      Type: AWS::DynamoDB::Table
      Properties:
        TableName: usersTable
        AttributeDefinitions:
          - AttributeName: email
            AttributeType: S
```

```yaml
# serverless.yml

service: users

provider:
  name: aws
  runtime: nodejs6.10
  stage: dev

functions:
  usersCreate:
    handler: users.create
    events:
      - http: post users/create

resources:
  Resources:
    usersTable:
      Type: AWS::DynamoDB::Table
      Properties:
        TableName: usersTable
        AttributeDefinitions:
          - AttributeName: email
            AttributeType: S
```

Microservice



```yaml
# serverless.yml

service: users

provider:
  name: aws
  runtime: nodejs6.10
  stage: dev
  STAGE: ${opt:stage, self:provider.stage}

functions:
  usersCreate:
    handler: users.create
    events:
      - http: post users/create

resources:
  Resources:
    usersTable:
      Type: AWS::DynamoDB::Table
      Properties:
        TableName: usersTable
        AttributeDefinitions:
          - AttributeName: email
            AttributeType: S
```

```
# serverless.yml

service: users

provider:
  name: aws
  runtime: nodejs6.10
  stage: dev
  STAGE: ${opt:stage, self:provider.stage}

functions:
  usersCreate:
    handler: users.create
    events:
      - http: post users/create

resources:
  Resources:
    usersTable:
      Type: AWS::DynamoDB::Table
      Properties:
        TableName: usersTable
        AttributeDefinitions:
          - AttributeName: email
            AttributeType: S
```

- **Service info as ENV vars**

```yaml
# serverless.yml

service: users

provider:
  name: aws
  runtime: nodejs6.10
  stage: dev
  STAGE: ${opt:stage, self:provider.stage}

functions:
  usersCreate:
    handler: users.create
    events:
      - http: post users/create

resources:
  Resources:
    usersTable:
      Type: AWS::DynamoDB::Table
      Properties:
        TableName: usersTable
        AttributeDefinitions:
          - AttributeName: email
            AttributeType: S
```

- Service info as ENV vars
- Inject in logs

```python
import observatory
obs = observatory.Tracker()

obs.track('search_made', {
    'query': event.query,
    'n_results': len(resp['data']),
    'user_id': user_item.id
})
```

After

- Rapid dev
- Source controlled
- Log enrichment

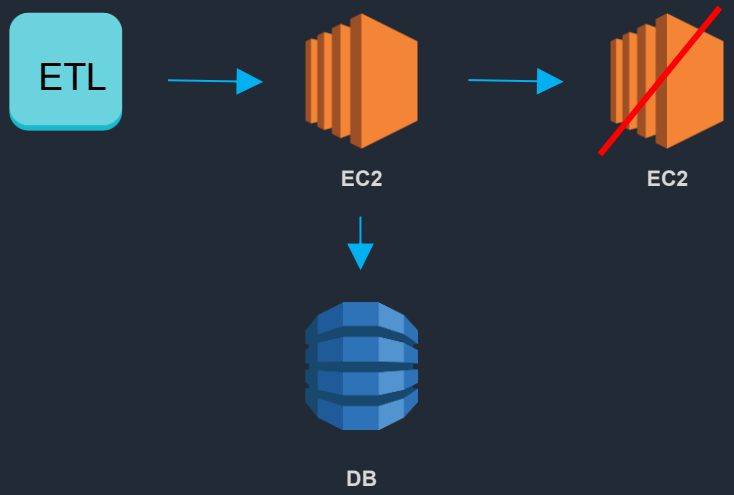
4. Server time == $$$

Before

- **Bursty**
- **ETL servers idle**

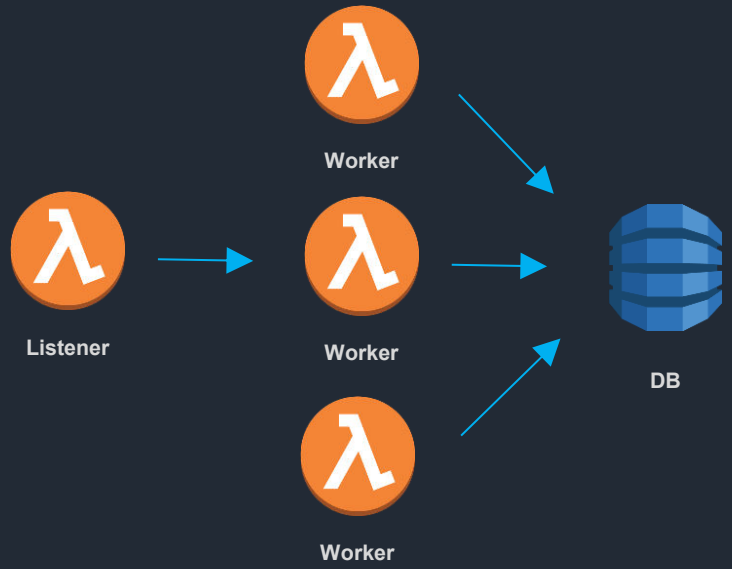
Transient Resources

- **Pipeline:**
  - **Spin up EC2**
  - **Terminate**



FAAS Resources

- Pipeline:
  - Discretize work
  - Lambda fleet
  - Inherently transient



After

- **Faster**
- **Cheaper**
- **Highly scalable ETL**

Distributed Data Engineering - Lessons

1. Metrics
2. Logging
3. Frameworks
4. Serverless ETL


Skywatch