SUMMIT

Toronto

Automating SFTP Workflows

Daniel Wierdsma
DevOps Engineer @ Tulip
Daniel.wierdsma@tulip.com

aws SUMMIT

Agenda

What is Tulip?

A Little Background

What Did We Build?

How Did We Improve On It?

Demo

Future Plans

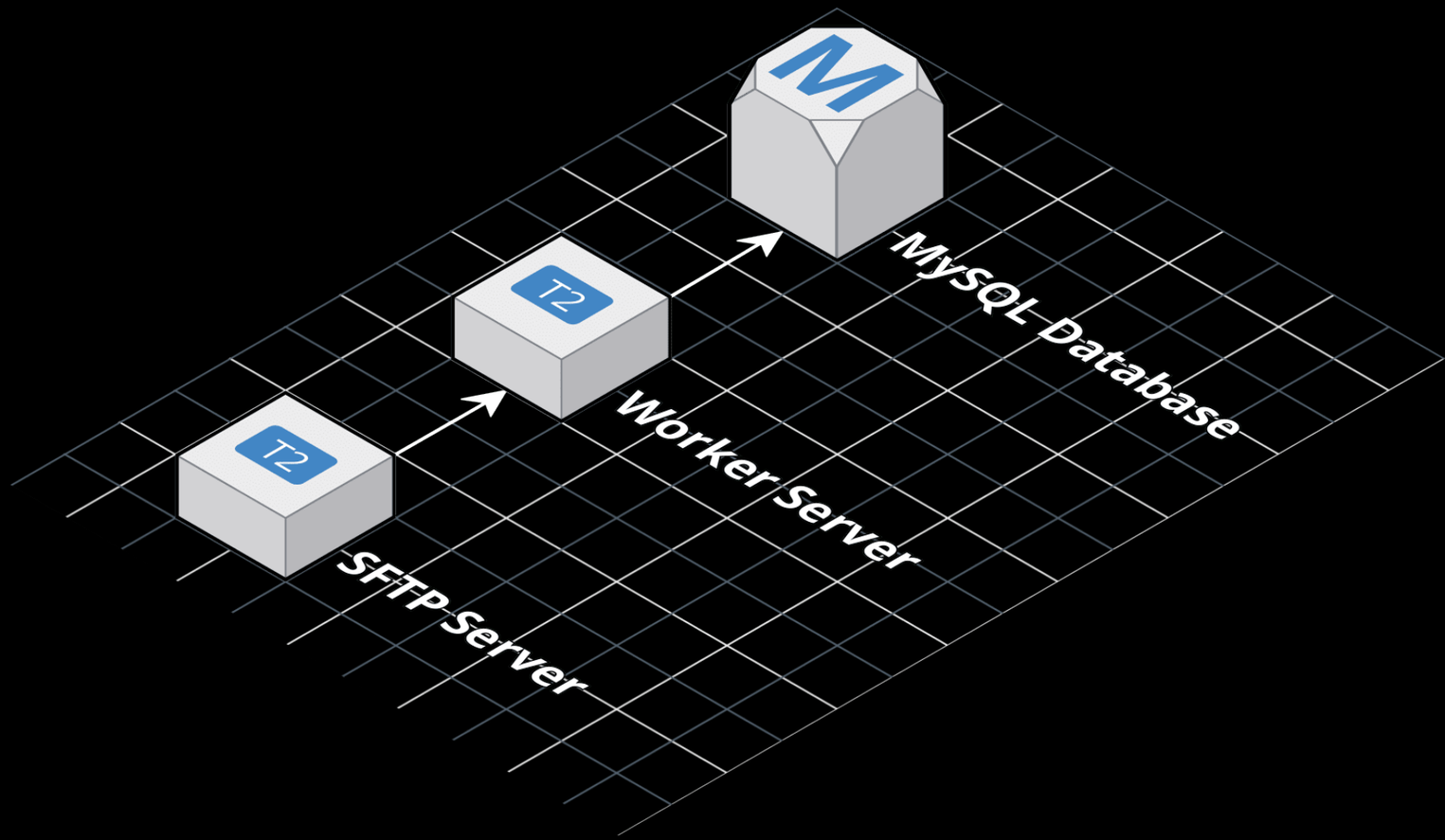
What is Tulip?

- SaaS Provider
- Empowering Sales Associates
- Cloud Backend

A Little Background

What We Had Before

- Fairly Basic ETL Pipeline
- Pull Files To Worker Servers
- Run Importers On Our Files



Pros

Simple Process/Design

Fairly easy to understand and build upon

Everything Was In One Place

Everything lived on our worker Linux machines

Small Blast Radius

If something broke it would break only one customers imports

Cons

Poor Visibility

Had to go through a lot of logs that were hard to parse through.

Relied On EC2 Instances Too Much

We wanted to move to containerized imports

Cron Based Scheduling

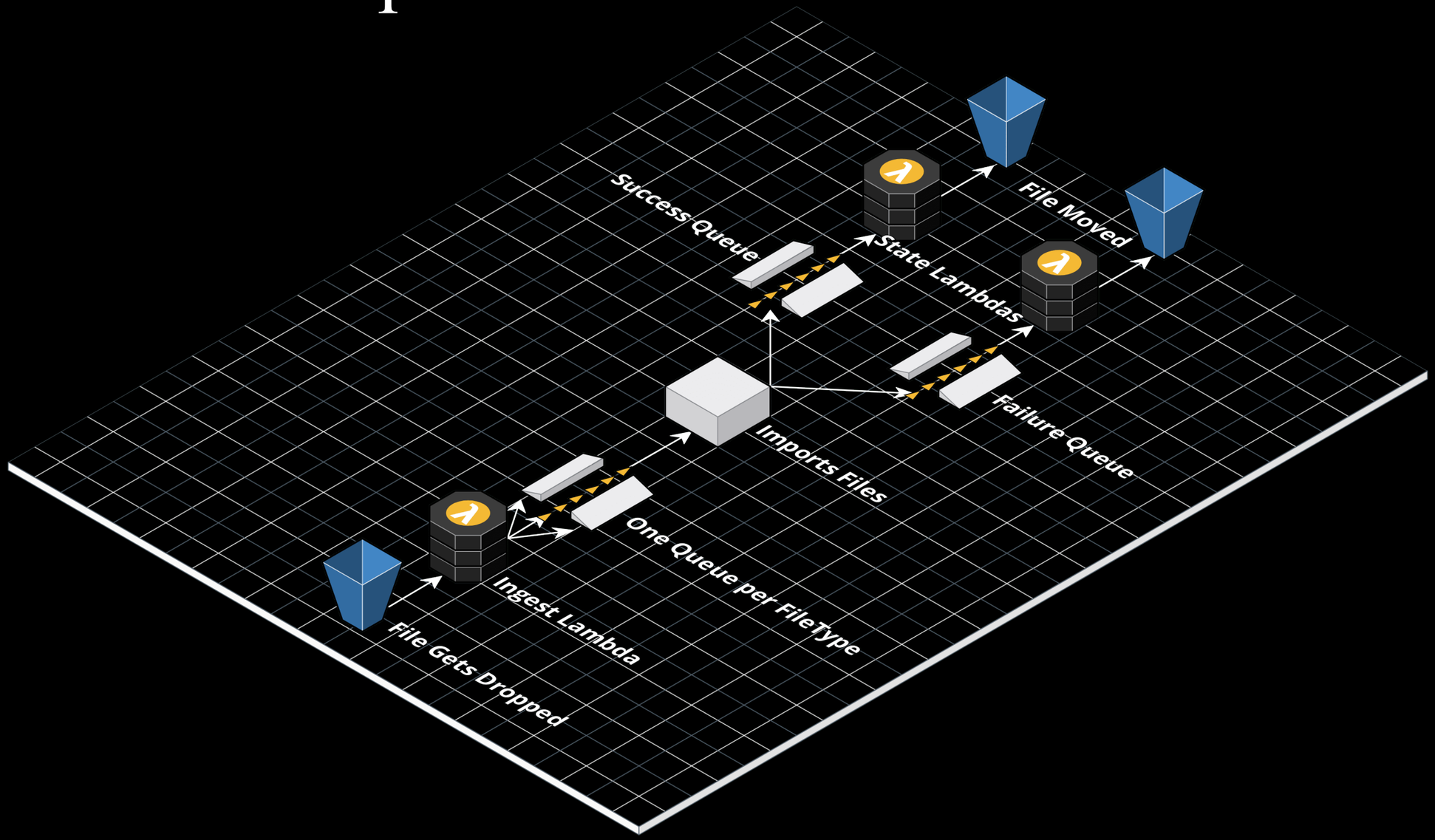
It would be better to trigger our importers automatically

What Did We Build?

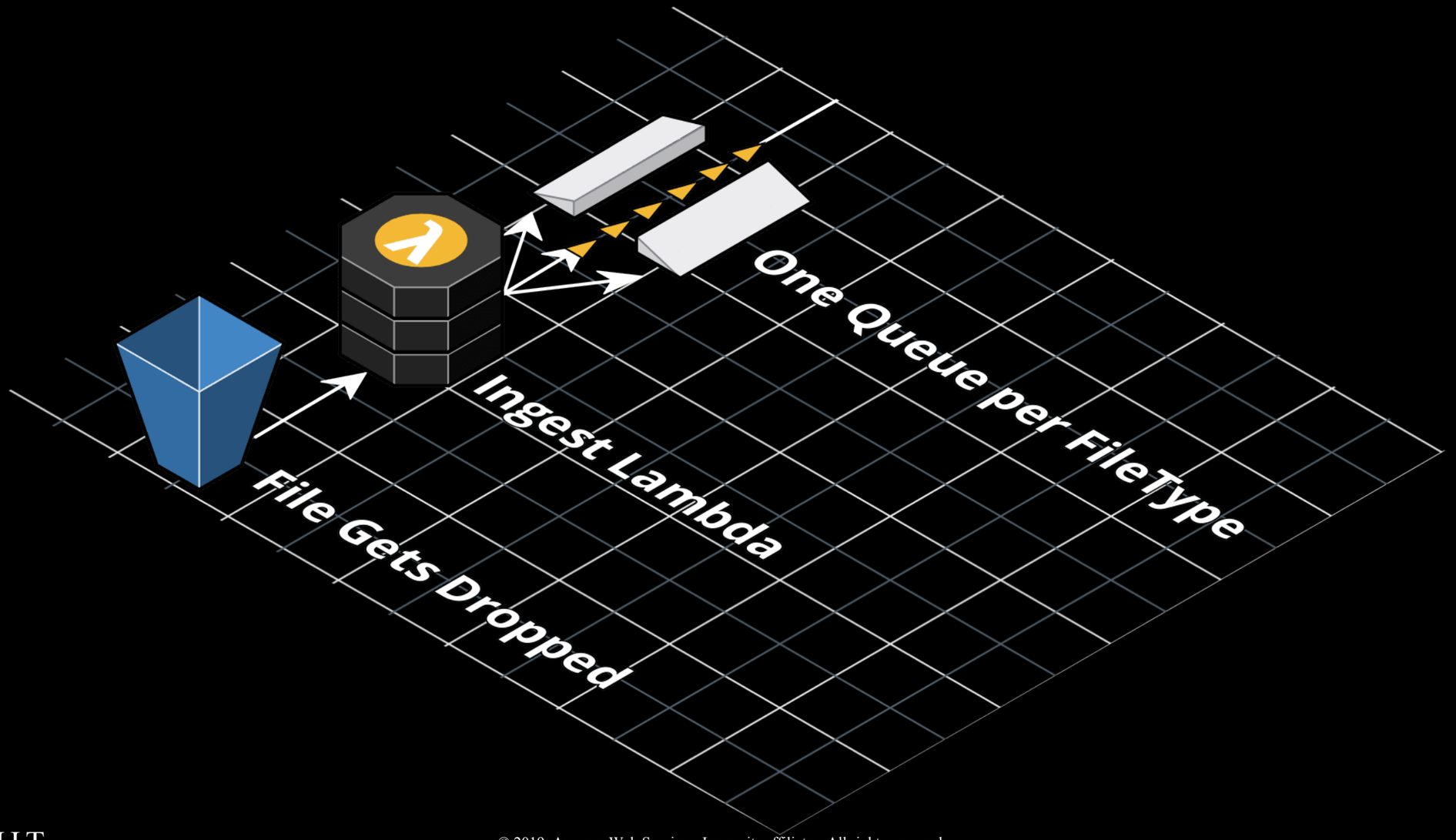
Why Did We Change

- Moving To AWS Transfer For SFTP
- Stop Relying On EC2 Instances
- Limit Work Done By Importers

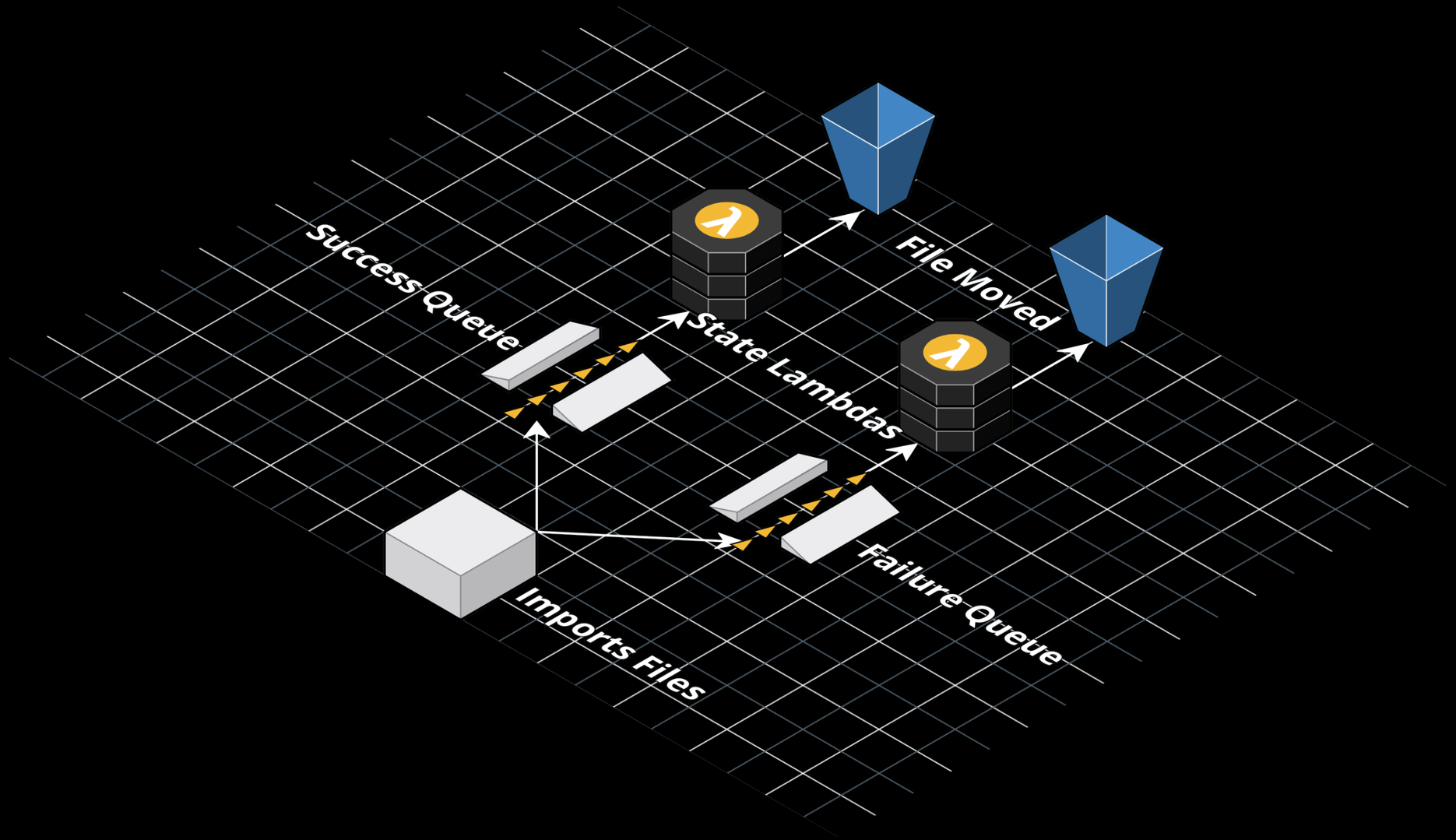
What We Came Up With - Overall



What We Came Up With - Ingest



What We Came Up With – State Updater



Takeaways

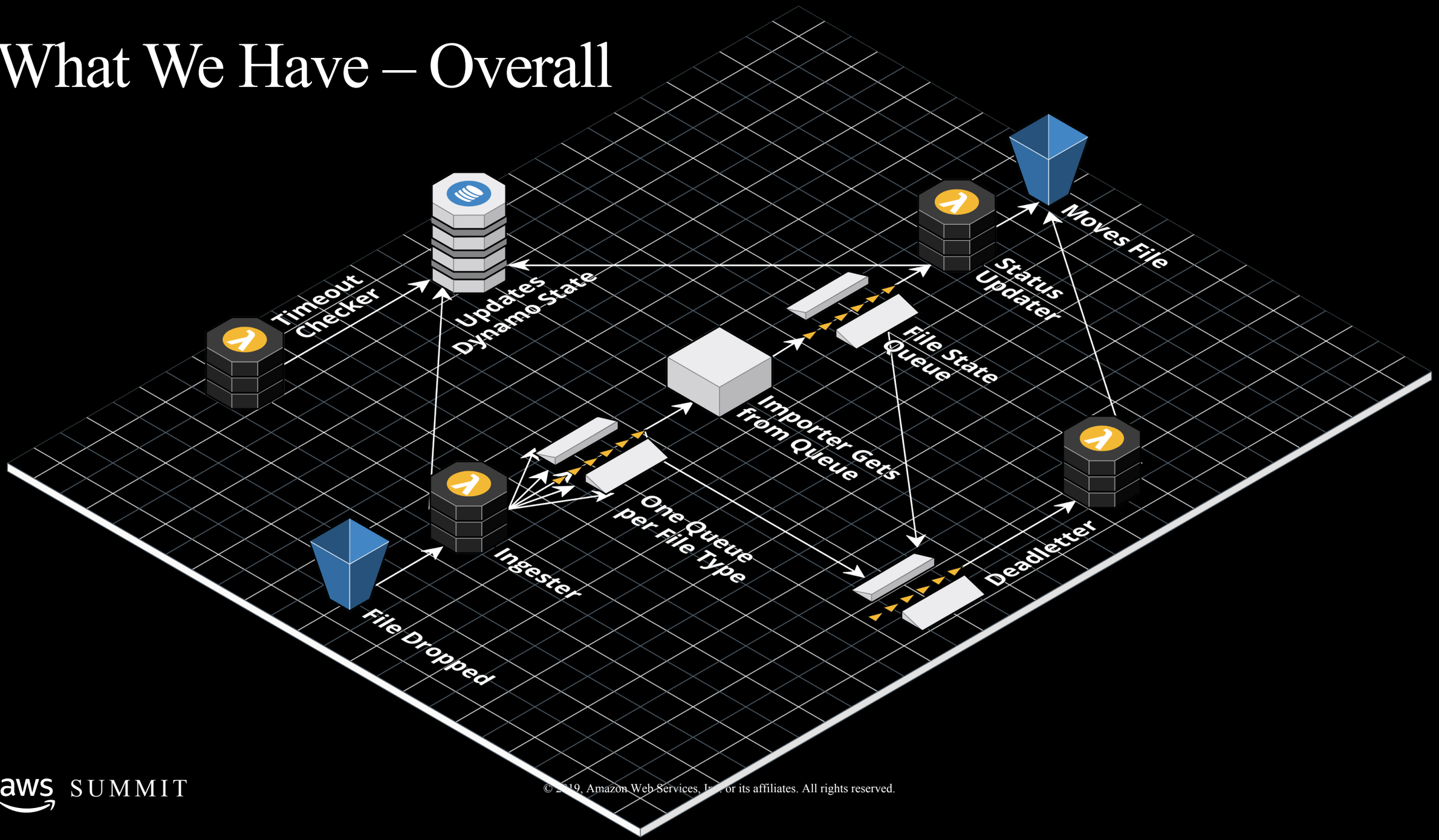
- Importers Are Only Responsible For Importing
- Added Config Files
- Lots Of Overhead
- Still Didn’t Have Great Visibility
- Good First Step; Could Be Better

How Did We Improve On It?

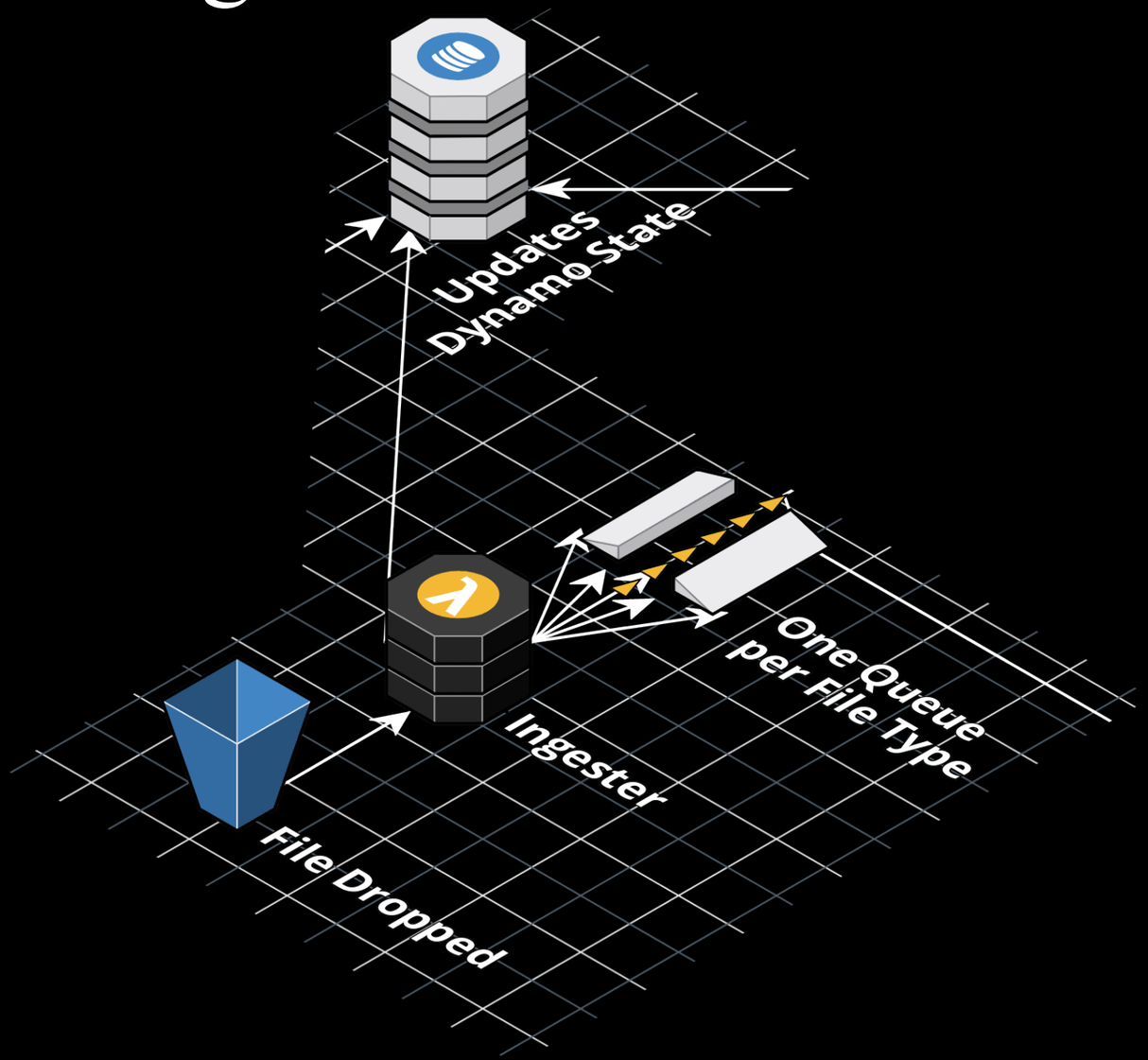
What We Improved

- We made the lambdas multi-tenant
  - Only one set of lambdas we don’t need to create a separate framework for each new client.
- Serverless Framework To Manage Lambdas
  - We now use serverless to deploy and track the state of our lambdas.
- Better Visibility
  - We created better logging, and Cloudwatch metrics.
- Added DynamoDB Tables
  - We added better data of the current state of files, constantly updating the associating records with their current state.

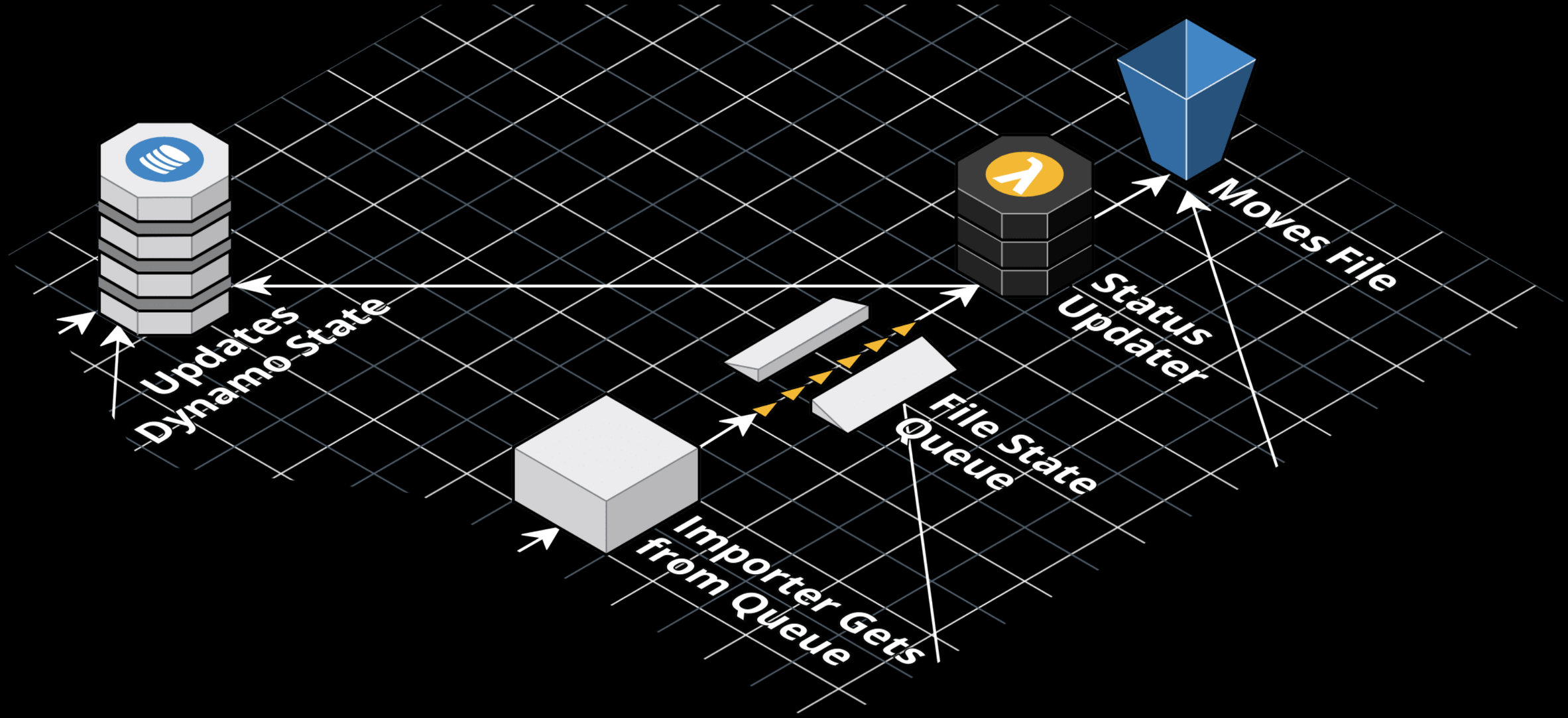
What We Have – Overall



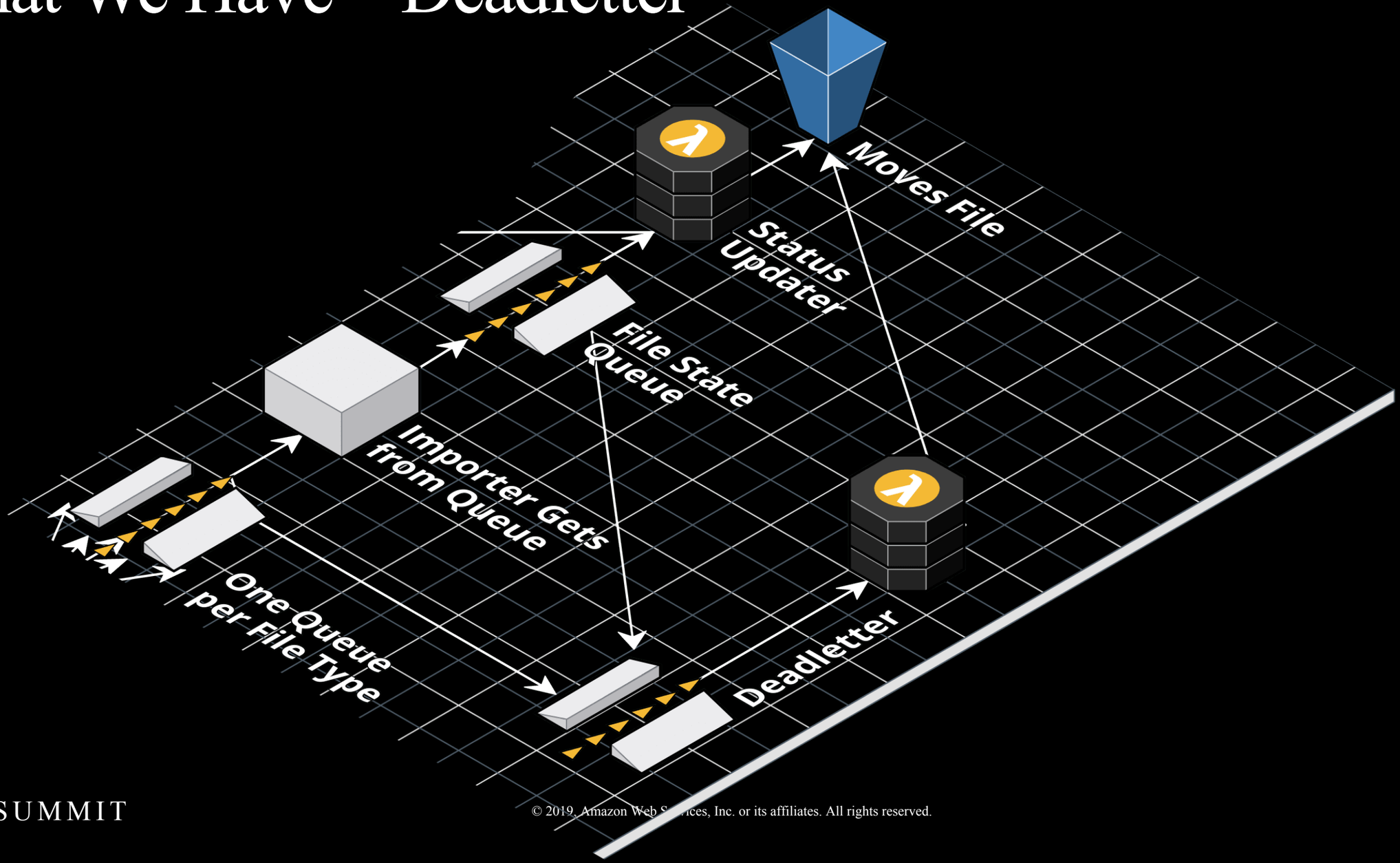
What We Have – Ingest



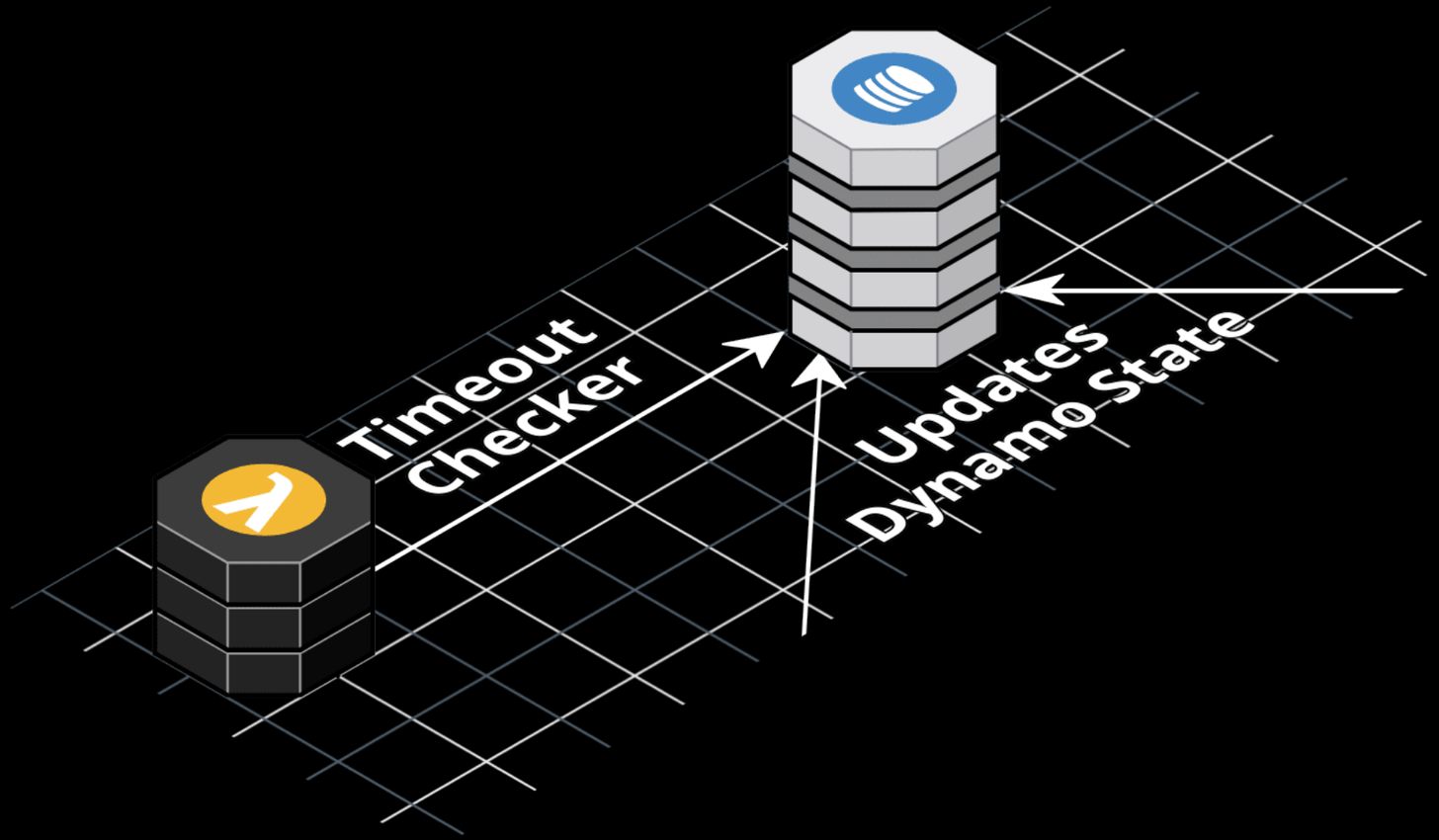
What We Have – State Updater



What We Have – Deadletter

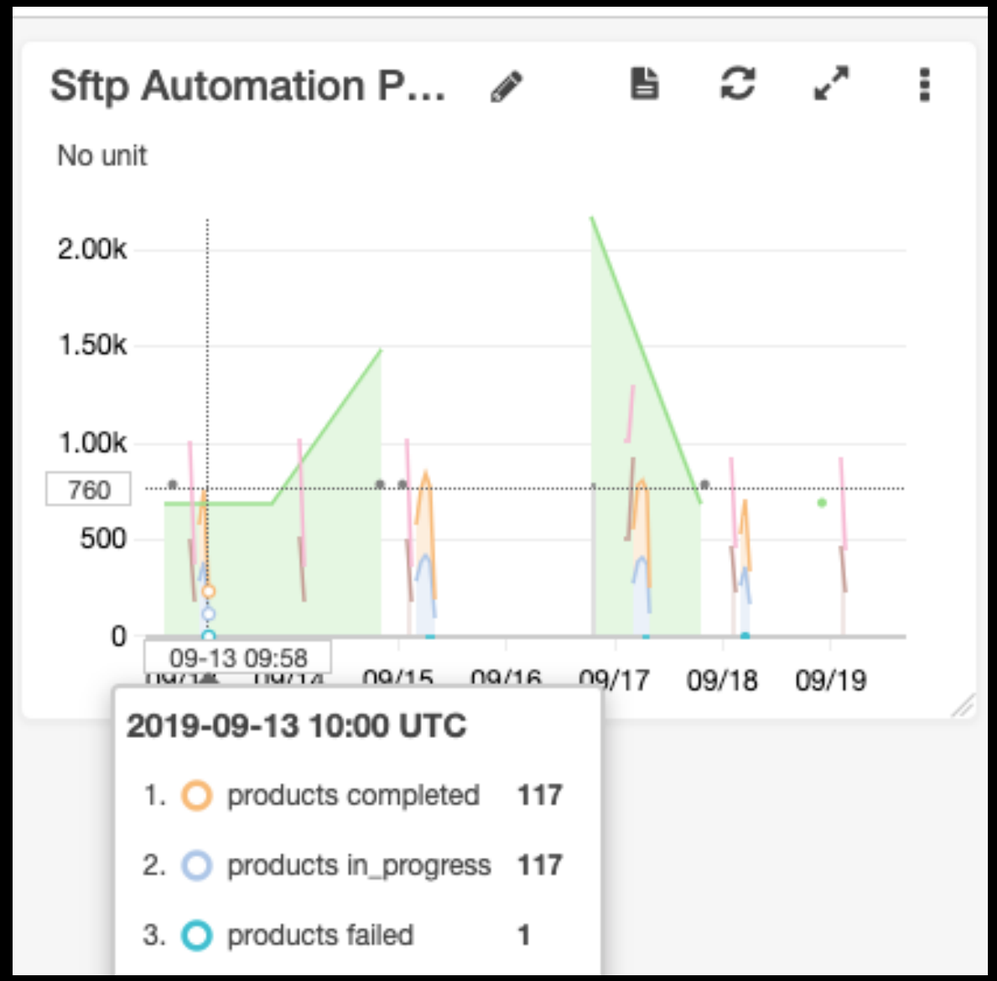


What We Have – Timeout Checker



CloudWatch Metrics

- Metrics For Each State And Tenant
  - Helps out parse on a per customer basis the progress of our imports.
- Created Dashboards Per Tenant
  - We created dashboards for each of our live customers that show the metrics that we have created on a handy graph.
- Data Can Be Used To Create Alarms
  - These metrics allows us to set up alarms that can let us know before our clients that something is wrong with imports.



Dynamo DB Records

- Provide State For Specific Files
  - In our Demo we have a shot where we show the date a file was dropped, current state and other details.
- On-Demand Read/Write Provisioning
  - This allows us to not worry about throttling our lambdas that could have many running at once.
- Supports Timeouts
  - This is good for determining whether our importers got in to a bad state or if they are running at all.

Serverless Framework

```
service: sftp-automation

provider:
  name: aws
  runtime: python3.7
  region: us-east-1

functions:
  ingester:
    handler: src/ingester/lambda_function.lambda_handler
    role: SftpAutomationRole
    events:
      - s3:
        bucket: ${self:custom.s3Bucket}

resources:
  Resources:
    SftpAutomationRole:
      <IAM Permissions Go Here>
```

Pros

Much Better Logging

Easy to find logs and metrics

Allows Us To Containerize

We can now containerize and standardize our imports easier

Less Overhead

If something broke it would break only one customers imports

Cons

More Services

Had to go through a lot of logs that were hard to parse through.

Large Blast Radius

Since we have a multi-tenant service any outage affects all customers

Still Cron Based Scheduling

It would be better to trigger our importers automatically

Demo

```
[~/workspace/infra/sftp-automation/serverless (develop)]$ make deploydev
Deploying dev stage using serverless tool
Serverless: Packaging service...
Serverless: Excluding development dependencies...
Serverless: Excluding development dependencies...
Serverless: Excluding development dependencies...
Serverless: Excluding development dependencies...
Serverless: Installing dependencies for custom CloudFormation resources...
Serverless: Uploading CloudFormation file to S3...
Serverless: Uploading artifacts...
Serverless: Uploading service ingester.zip file to S3 (5.55 KB)...
Serverless: Uploading service timeout-checker.zip file to S3 (3.95 KB)...
Serverless: Uploading service status-updater.zip file to S3 (7.42 KB)...
Serverless: Uploading service deadletter.zip file to S3 (4.52 KB)...
Serverless: Uploading custom CloudFormation resources...
Serverless: Validating template...
Serverless: Updating Stack...
Serverless: Checking Stack update progress...
.............
Serverless: Stack update finished...
```

Future Plans

aws SUMMIT

How To Make It Even Better

- Change How Importers Run Files
  - We want to make each message sent to importers a record instead of a file location
- Rework The Imports Themselves
  - Create a configurable ETL process to enforce data standards
- Change How Our Importers Are Hosted
  - Either containerized, or Serverless frameworks

Thank you!

Daniel Wierdsma
Daniel.Wierdsma@tulip.com
https://github.com/d-wierdsma/sftp-automation-talk

aws SUMMIT

Please complete the session survey.