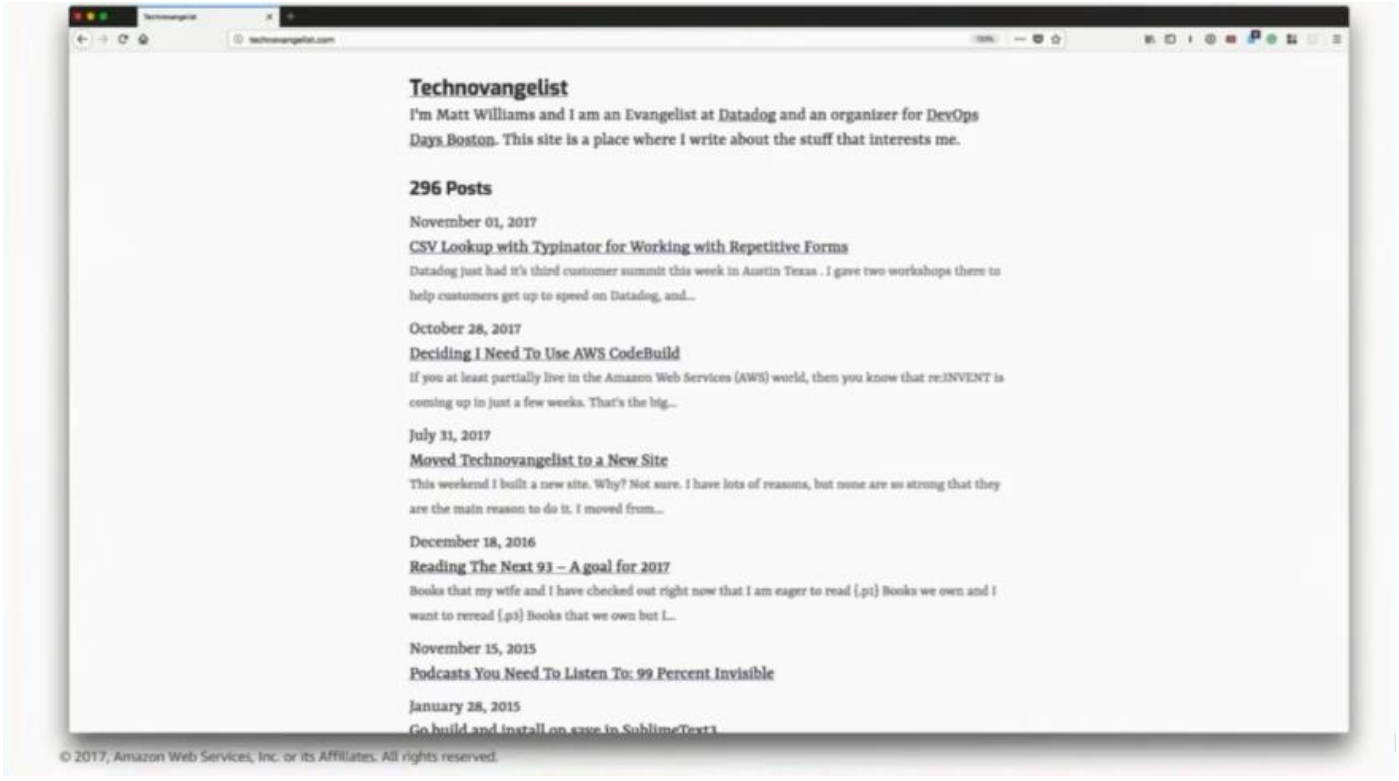
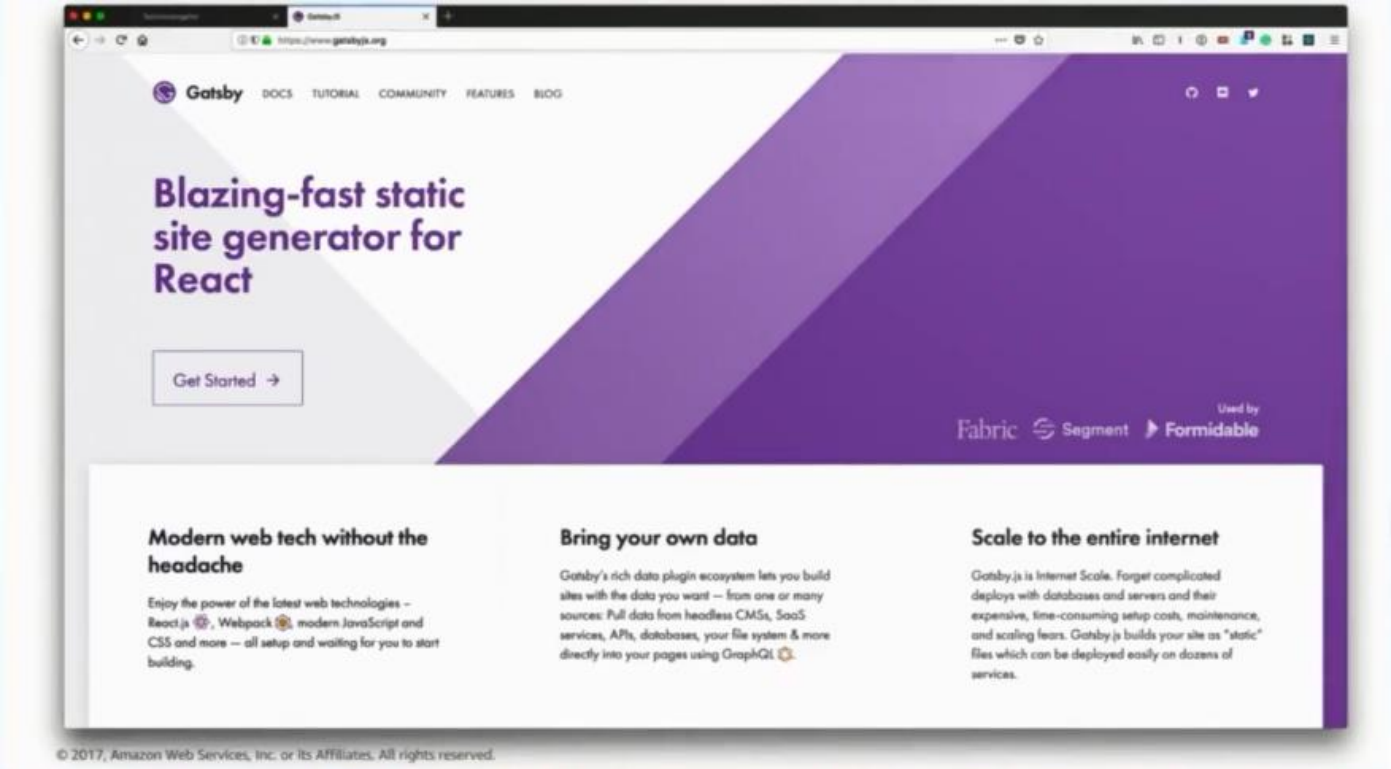
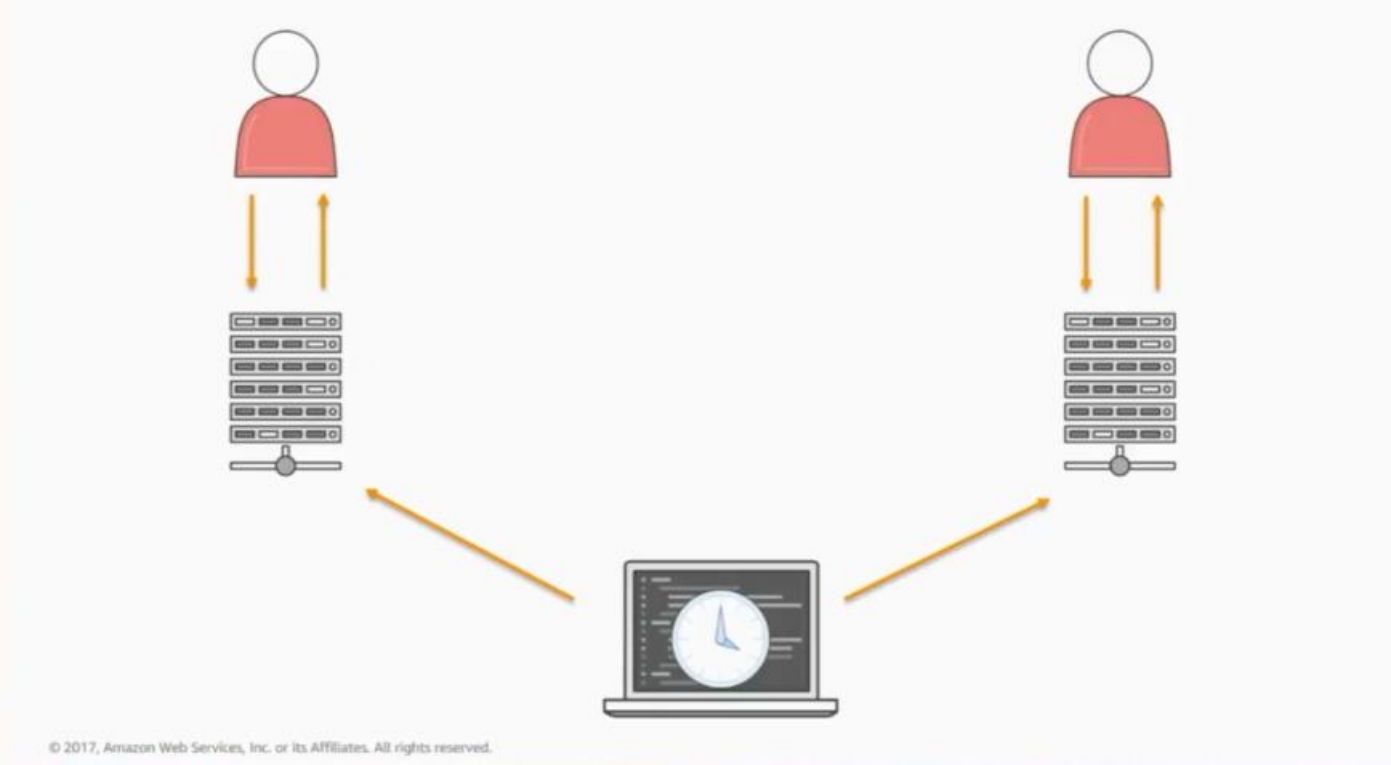

AWS re:Invent
Best Practices for Orchestrating AWS Lambda Workloads
Matthew Williams
Evangelist @ Datadog
AWS re:Invent
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
aws

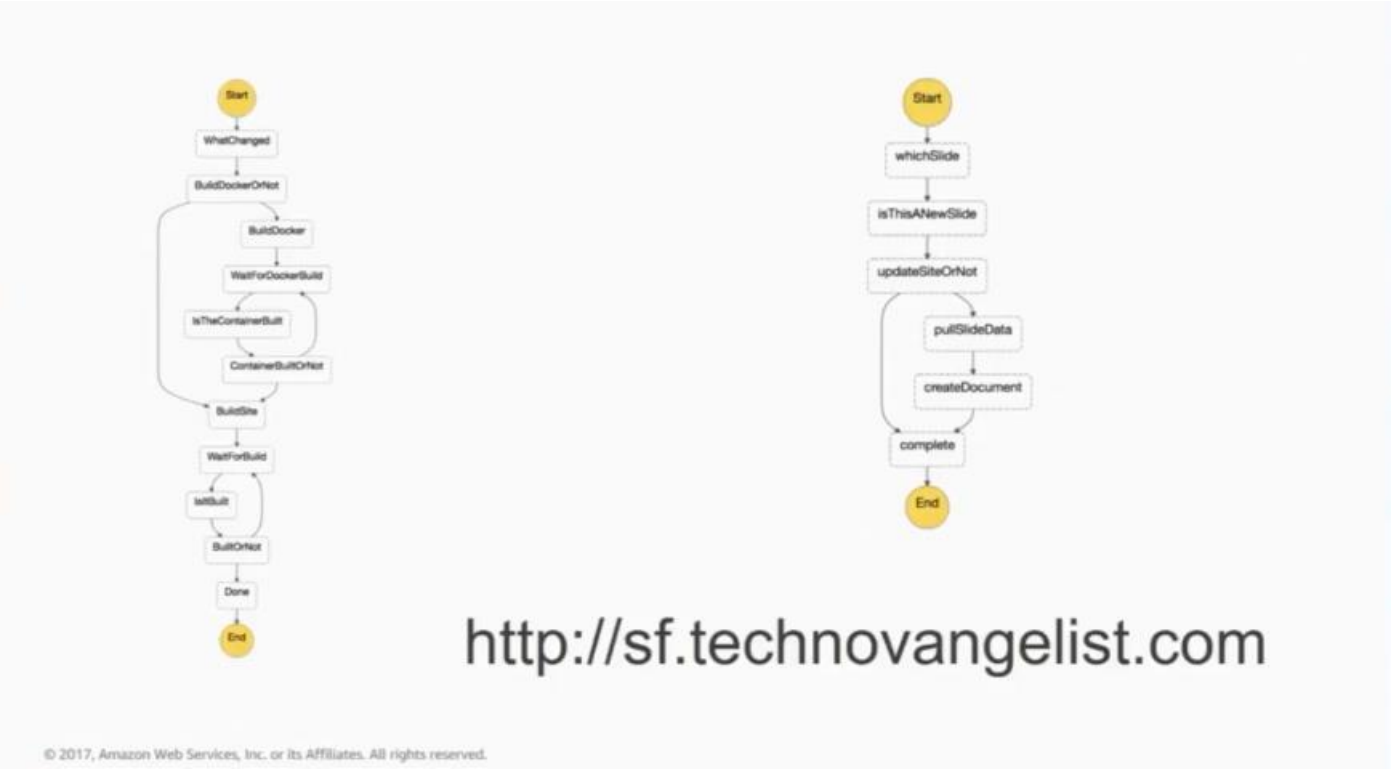

Technovangelist
I'm Matt Williams and I am an Evangelist at Datadog and an organizer for DevOps Days Boston. This site is a place where I write about the stuff that interests me.
296 Posts
November 01, 2017
CSV Lookup with Typinator for Working with Repetitive Forms
Datadog just had it's third customer summit this week in Austin Texas . I gave two workshops there to help customers get up to speed on Datadog, and...
October 28, 2017
Deciding I Need To Use AWS CodeBuild
If you at least partially live in the Amazon Web Services (AWS) world, then you know that re:INVENT is coming up in just a few weeks. That's the big...
July 31, 2017
Moved Technovangelist to a New Site
This weekend I built a new site. Why? Not sure. I have lots of reasons, but none are so strong that they are the main reason to do it. I moved from...
December 18, 2016
Reading The Next 93 – A goal for 2017
Books that my wife and I have checked out right now that I am eager to read {.p1} Books we own and I want to reread {.p1} Books that we own but I...
November 15, 2015
Podcasts You Need To Listen To: 99 Percent Invisible
January 28, 2015
Go build and install on save in SublimeText3
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



Now using GatsbyJS hosted on S3 for building out my blog that converts our JS, CSS, HTML assets into a static site



We don’t need to keep a fixed EC2 instance when



We are going to be talking about 2 step functions that are used to build our static sites

Who is Datadog

SaaS-based Monitoring & Analytics
Infrastructure, APM, Logs
Open Source Agent
Trillions of data
points per day

Booth 109 in Aria
Booth 1021 in Venetian
We are hiring!!

AWS re:Invent

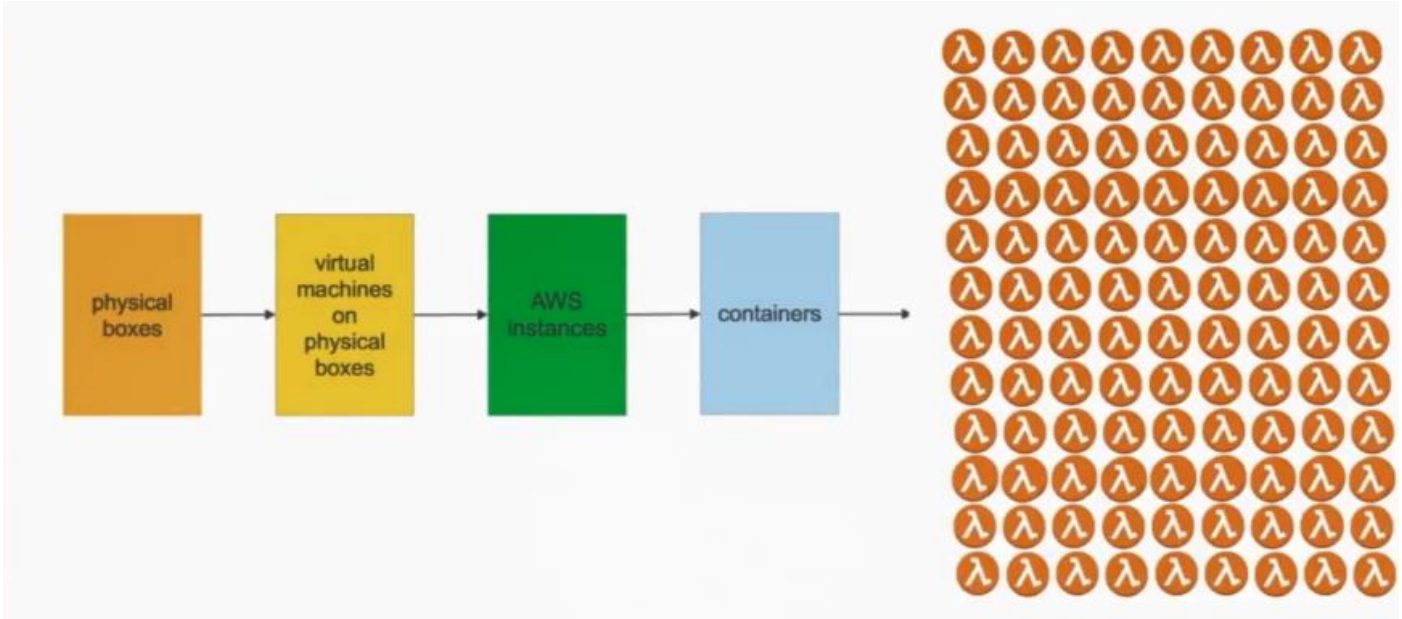
aws

...but AWS Lambda != Amazon EC2

...but AWS Lambda != application

...applications are complicated

...Lambdas should respect the single responsibility principle...



Other things learned over the last year

Serverless is about automation

Serverless is about event-driven computing

Scalability is different

we can create many serverless functions

but how do we orchestrate them?

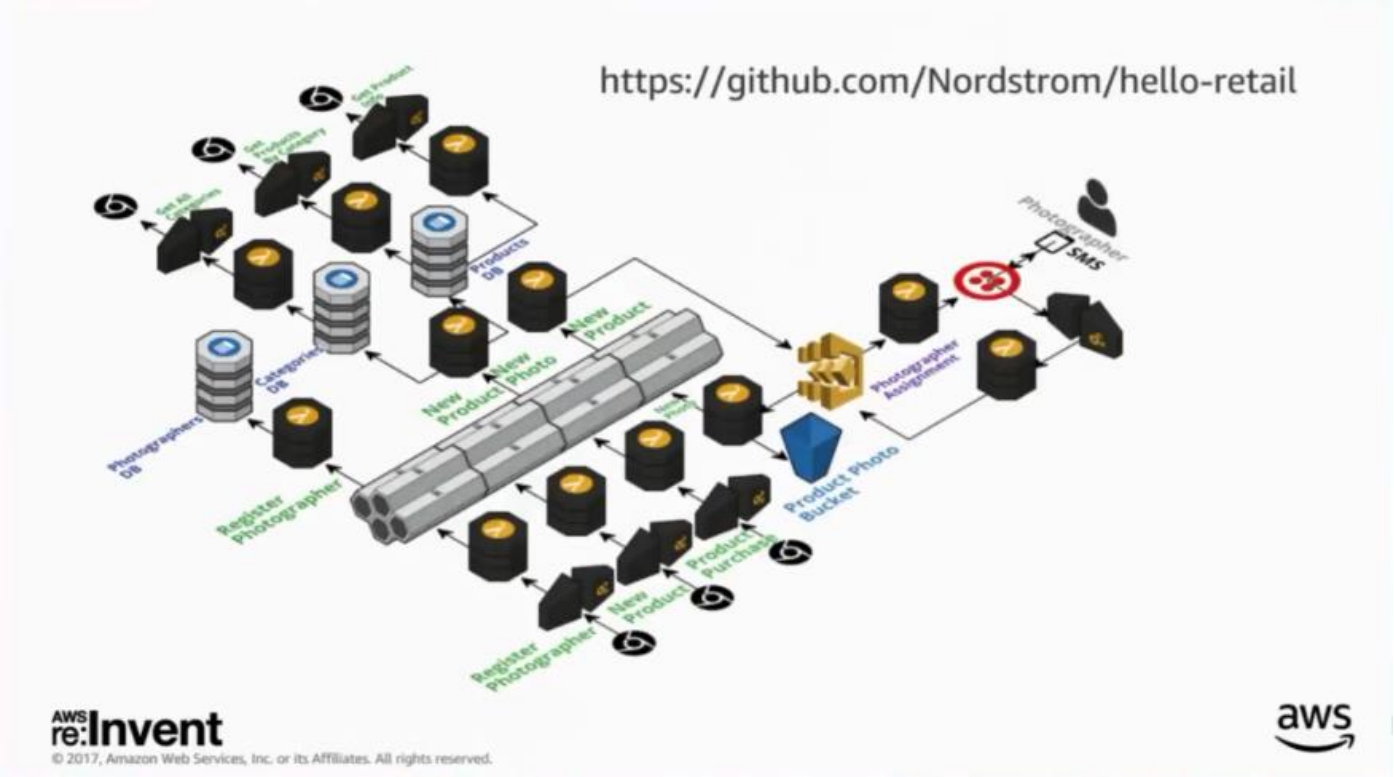
Event/Message/Database as Traffic Cop

Amazon Kinesis

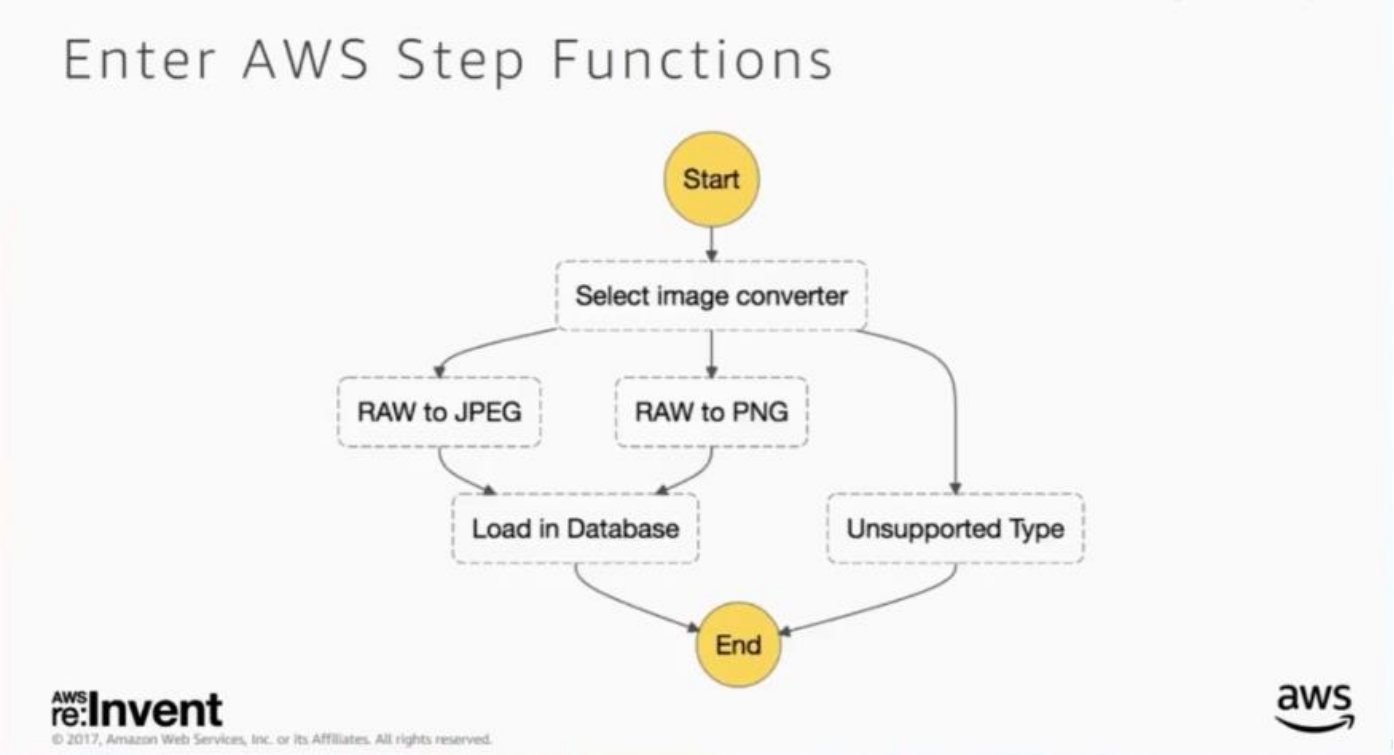
Amazon SQS

Amazon DynamoDB

The concept of the Traffic Cop is something that a lambda can report back to after it does its work. Then the next lambda can look into the Traffic Cop to determine whether it should do its work or not.



This is a very good example of the Traffic Cop concept. You can try this out using the GitHub code and link above



Now we can orchestrate several lambdas that each do their individual work and then hand off the result to another lambda function using Step Functions

a step function is
a state machine

but what about SWF?

(Amazon Simple Workflow Service)

AWS Step Function Concepts

- State Machine
- Transitions
- Executions
- States
    - Task
    - Choice
    - Succeed/Fail
    - Pass
    - Wait
    - Parallel

You define the State Machine with the Amazon States Language

A diagram will be generated based on that structure

AWS Step Function States: Task

Resource *
ResultPath
Retry
Catch
TimeoutSeconds
HeartbeatSeconds

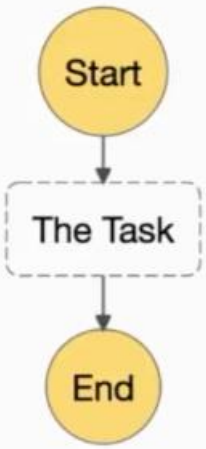
Type *
Next
End
Comment
InputPath
OutputPath

A task can be a Lambda or an 'activity'

The Type* is the task and the Resource* is the ARN for that lambda or activity.

State Example: Task

```
{
  "Comment": "A simple task example",
  "StartAt": "The Task",
  "States": {
    "The Task": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:REGION:ACCOUNT_ID:function:AwesomeTask",
      "End": true
    }
  }
}
```



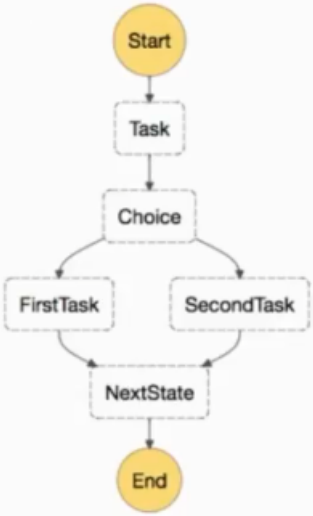
AWS Step Function States: Choice

Choices
Default

Type
Next
End
Comment
InputPath
OutputPath

State Example: Choice

```
    "Choice": {
      "Type" : "Choice",
      "Choices": [
        {
          "Variable": "$.foo",
          "NumericEquals": 1,
          "Next": "FirstTask"
        },
        {
          "Variable": "$.foo",
          "NumericEquals": 2,
          "Next": "SecondTask"
        }
      ]
    },
```



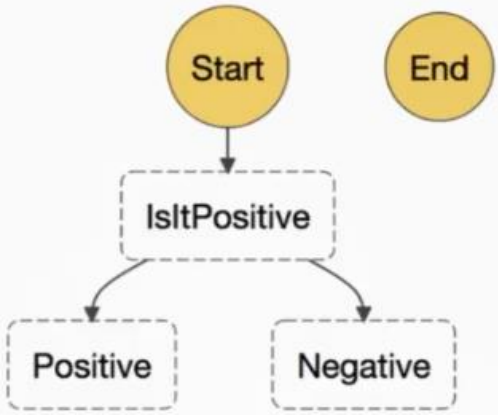
AWS Step Function States: Succeed / Fail

Cause
Error

Type
Next
End
Comment
InputPath
OutputPath

State Example: Succeed / Fail

```
7    "Choices": [
8      {
9        "Variable": "$.foo",
10       "NumericGreaterThan": 0,
11       "Next": "Positive"
12     }
13   ],
14   "Default": "Negative"
15 },
16
17 "Positive": {
18   "Type" : "Succeed"
19 },
20
21 "Negative": {
22   "Type": "Fail",
23   "Error": "DefaultStateError",
24   "Cause": "Its not positive!"
25 }
```



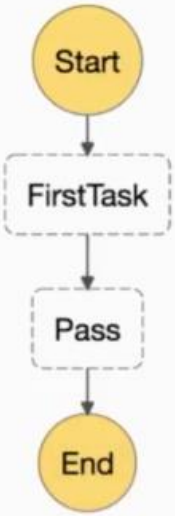
AWS Step Function States: Pass

Result
ResultPath

Type
Next
End
Comment
InputPath
OutputPath

State Example: Pass

```
{
  "Comment": "A simple pass",
  "StartAt": "FirstTask",
  "States": {
    "FirstTask": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:REGION:ACCOUNT_ID:function:AwesomeTask",
      "Next": "Pass"
    },
    "Pass": {
      "Type": "Pass",
      "Result": "Wow, this was exciting",
      "ResultPath": "ExtraDetail",
      "End": true
    }
  }
}
```



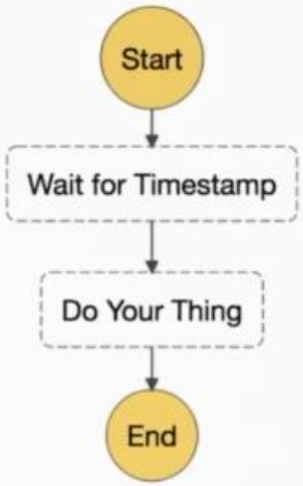
AWS Step Function States: Wait

Seconds
Timestamp
SecondsPath
TimestampPath

Type
Next
End
Comment
InputPath
OutputPath

State Example: Wait

```
{
  "Comment": "A simple wait example",
  "StartAt": "Wait for Timestamp",
  "States": {
    "Wait for Timestamp": {
      "Type": "Wait",
      "TimestampPath": "$.trigger_date",
      "Next": "Do Your Thing"
    },
    "Do Your Thing": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:REGION:ACCOUNT_ID:function:DoTheThing",
      "End": true
    }
  }
}
```



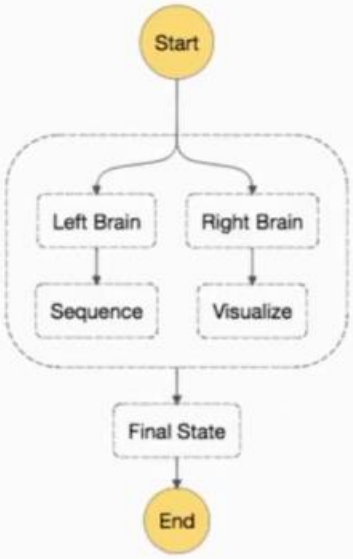
AWS Step Function States: Parallel

Branches
ResultPath
Retry
Catch

Type
Next
End
Comment
InputPath
OutputPath

State Example: Parallel

```
"Parallel": {
  "Type": "Parallel",
  "Next": "Final State",
  "Branches": [
    {
      "StartAt": "Left Brain",
      "States": {
        "Left Brain": {
          "Type": "Wait",
          "Seconds": 2,
          "Next": "Sequence"
        },
        "Sequence": {
          "Type": "Task",
          "Resource": "arn:aws:lambda:REGION:ACCOUNT_ID:function:sequence",
          "End": true
        }
      }
    },
    {
      "StartAt": "Right Brain",
      "States": {
        "Right Brain": {
          "Type": "Pass",
          "Next": "Visualize"
```



These are the basics of using step functions

Many ways to build Step Functions

I used Serverless Framework

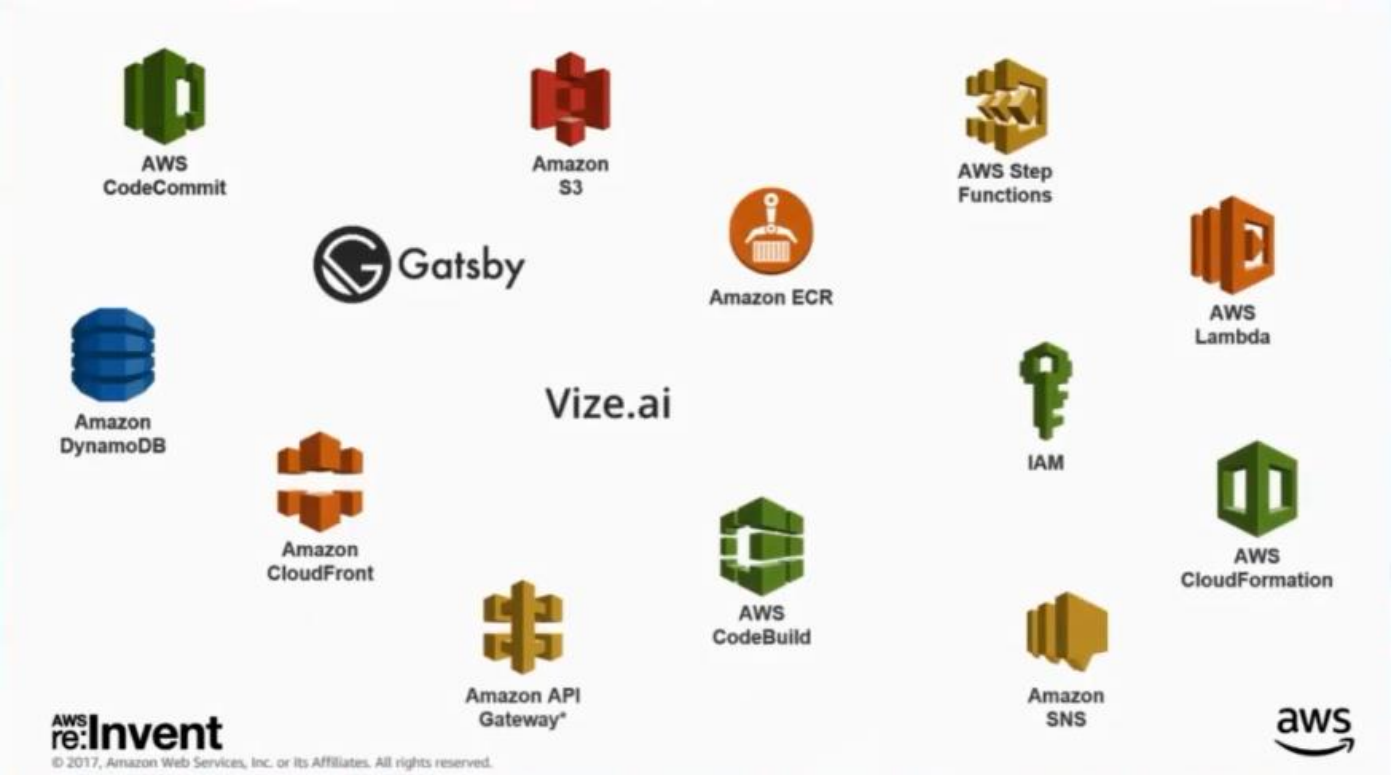
serverless

The ***Serverless framework*** has an add-on called ***Serverless Step Functions*** that make it really easy to build step functions and their integrations in a single file

Benefits of Serverless Framework

Define everything in a single file:

- IAM Role Statements
- Lambda Function Handlers
- Lambda Function Triggers
- Step Function States
- CloudFormation for other things



serverless.yml

```
service:
  name: mattw-reinvent2017-build-website


plugins:
  - serverless-pseudo-parameters
  - serverless-step-functions
  - serverless-webpack
custom:
  repoName: reinvent2017-website
```

serverless.yml cont'd

```
provider:
  name: aws
  runtime: nodejs6.10
  stage: ${opt:stage, 'dev'}
  region: ${opt:region, 'us-east-1'}
  iamRoleStatements:
    - Effect: "Allow"
      Action:
        - "states:StartExecution"
      Resource:
        - ${self:resources.Outputs.MyStateMachine.Value}
    - Effect: "Allow"
      Action:
        - "codecommit:GetCommit"
        - "codecommit:GetDifferences"
      Resource:
        - arn:aws:codecommit:${self:provider.region}:#{AWS::AccountId}:${...custom.repoName}
```

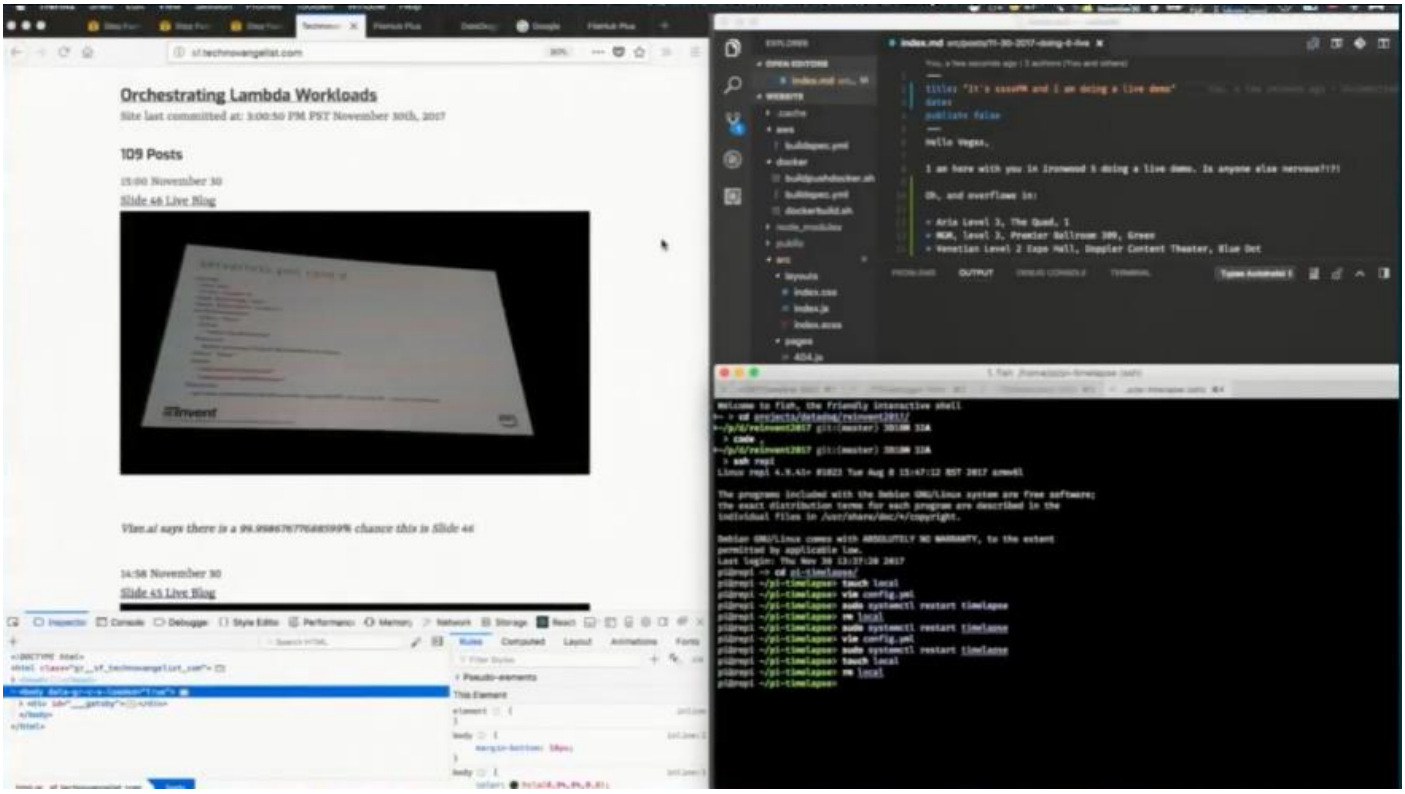

serverless.yml cont'd

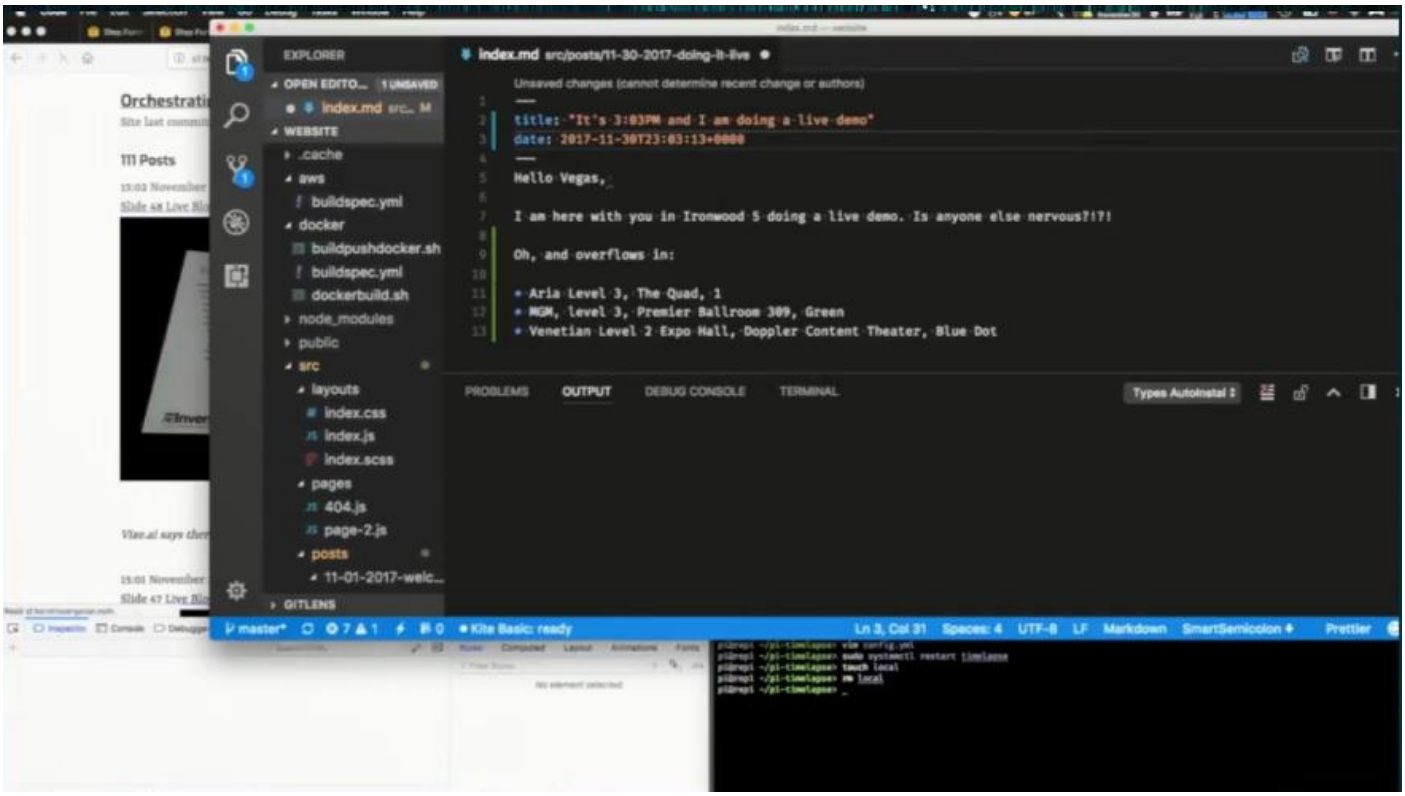
```
functions:
  buildSite:
    handler: handler.build
  whatChanged:
    handler: handler.whatChanged
  startStepFunction:
    handler: handler.startStepFunction
    events:
      - sns: arn:aws:sns:${...provider.region}:#{AWS::AccountId}:mattw-reinvent-websitebuild
    environment:
      statemachine_arn: ${self:resources.Outputs.MyStateMachine.Value}
```

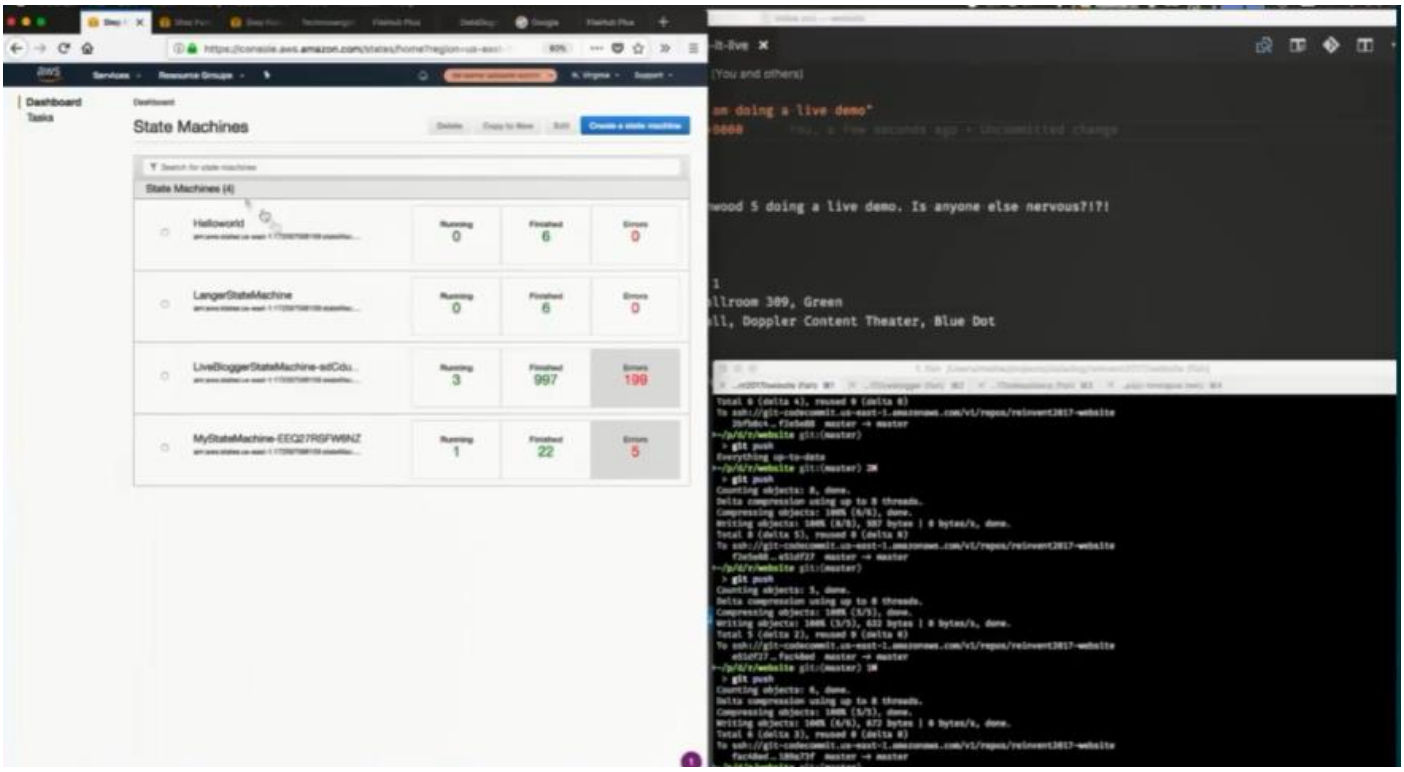
serverless.yml cont'd

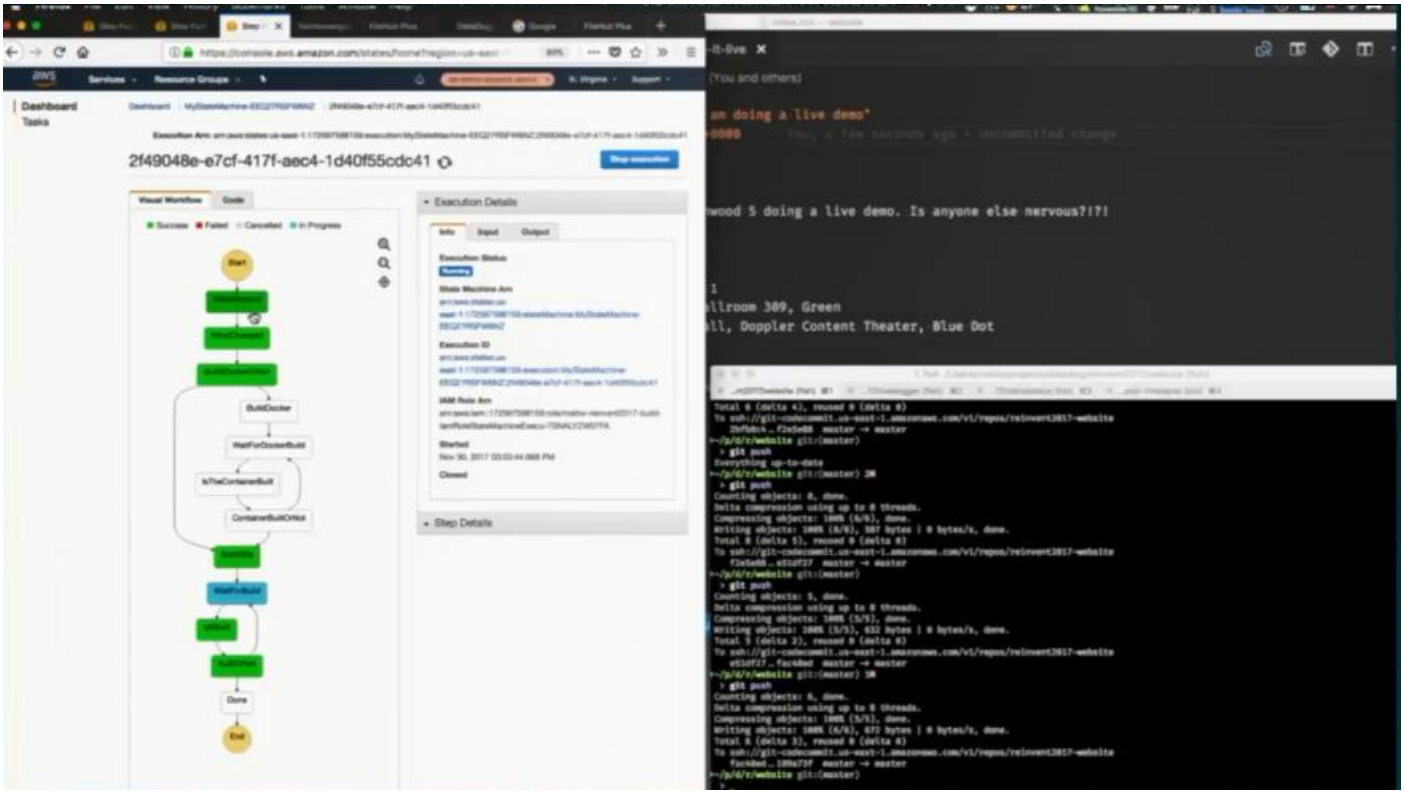
```
stepFunctions:
  stateMachines:
    FirstStateMachine:
      name: myStateMachine
      definition:
        StartAt: WhatChanged
        States:
          WhatChanged:
            Type: Task
            Resource: arn:aws:lambda:#{AWS::Region}:#{AWS::AccountId}\
          :function:${self:service}-${self:provider.stage}-whatChanged
            Next: BuildDockerOrNot
```

Demo

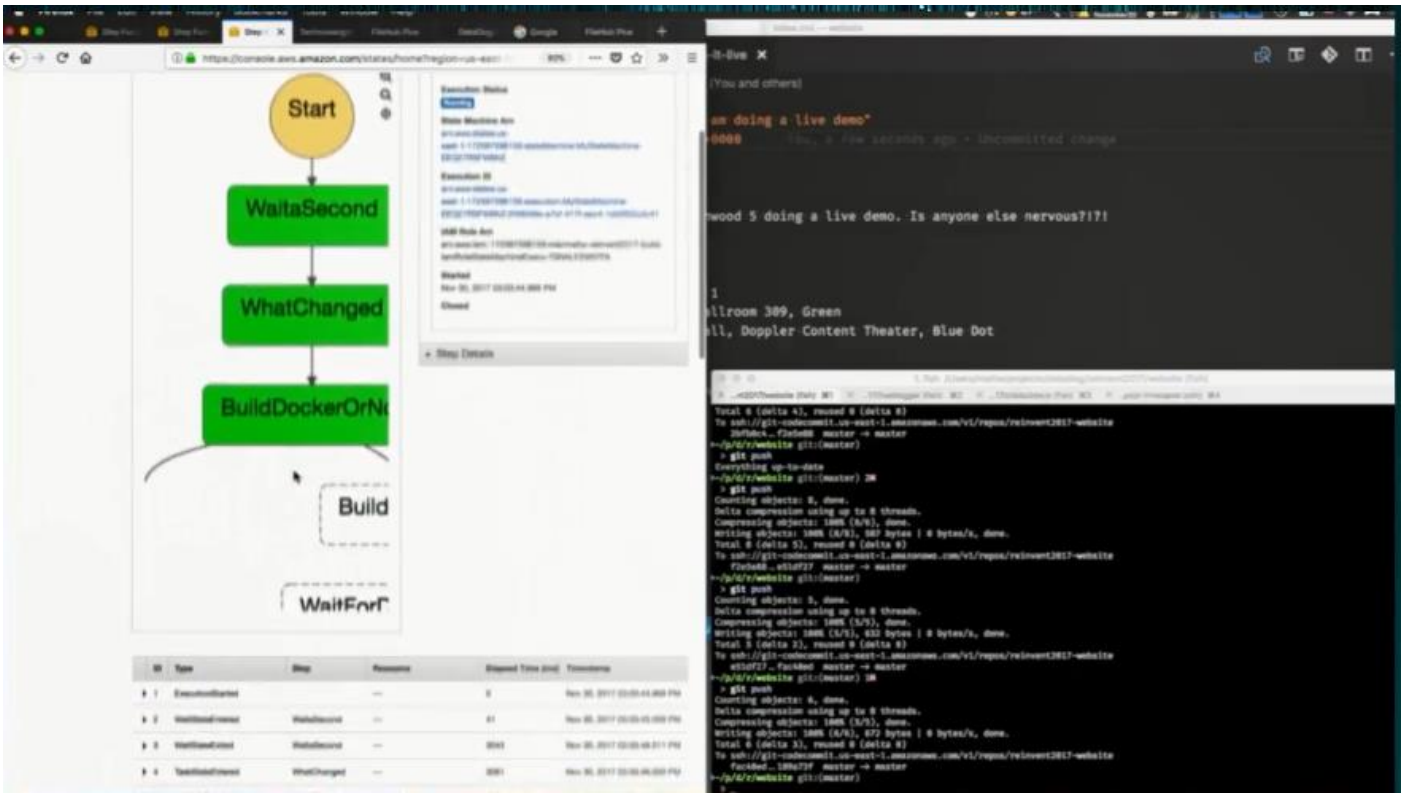

Orchestrating Lambda Workloads
Site last committed at: 3:00:50 PM PST November 30th, 2017
109 Posts
15:00 November 30
Slide 46 Live Blog
Vlan.ai says there is a 99.9986767704693998% chance this is Slide 46
14:58 November 30
Slide 45 Live Blog
Hello Vegas,
I am here with you in Ironwood 5 doing a live demo. Is anyone else nervous?!?!
Oh, and overflows in:
Aria Level 3, The Quad, 1
MGM, Level 3, Premier Ballroom 309, Green
Venetian Level 2 Expo Hall, Doppler Content Theater, Blue Dot

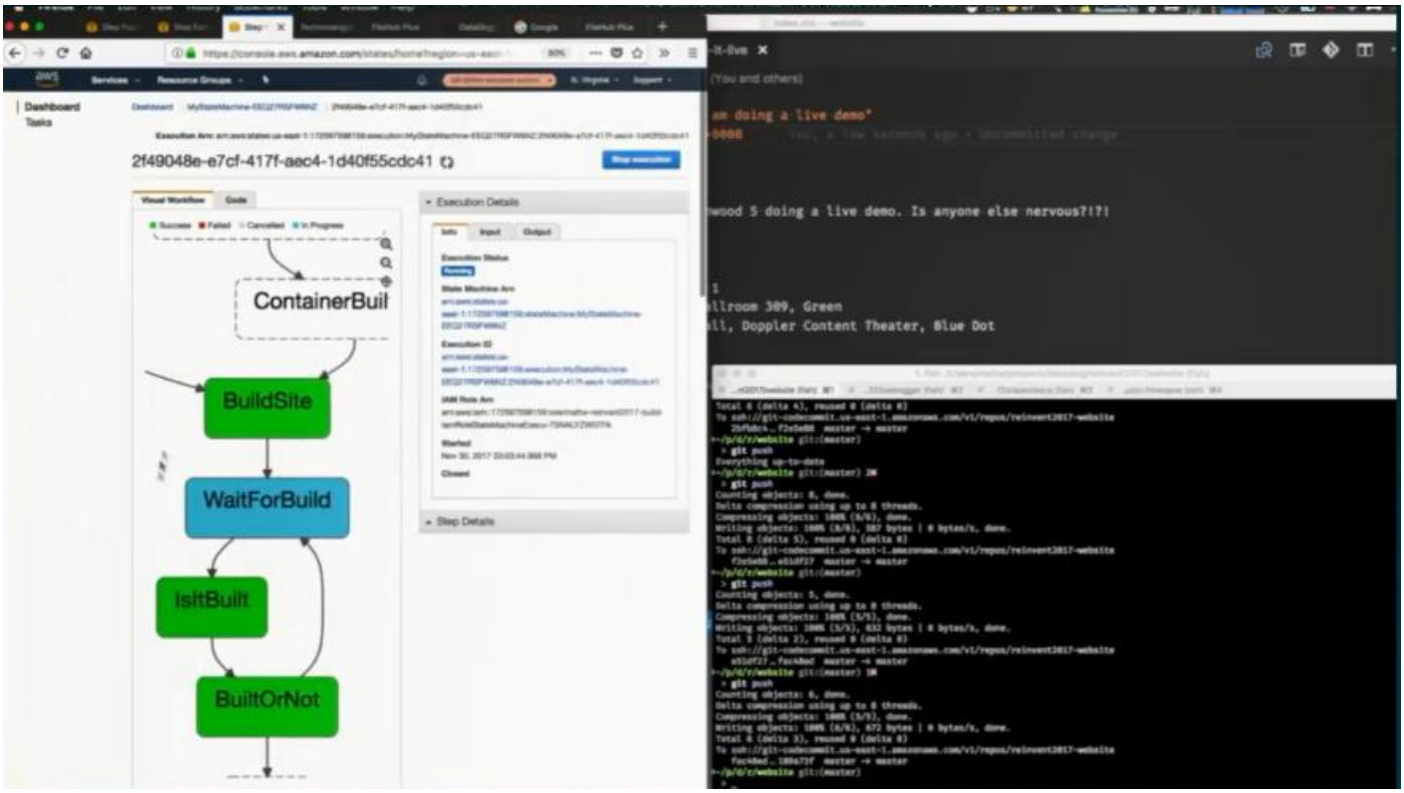

EXPLORER
index.md src/posts/11-30-2017-doing-it-live
Unsaved changes (cannot determine recent change or authors)
title: "It's 3:03PM and I am doing a live demo"
date: 2017-11-30T23:03:13+0000
Hello Vegas,
I am here with you in Ironwood 5 doing a live demo. Is anyone else nervous?!?!
Oh, and overflows in:
Aria Level 3, The Quad, 1
MGM, level 3, Premier Ballroom 309, Green
Venetian Level 2 Expo Hall, Doppler Content Theater, Blue Dot
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
111 Posts

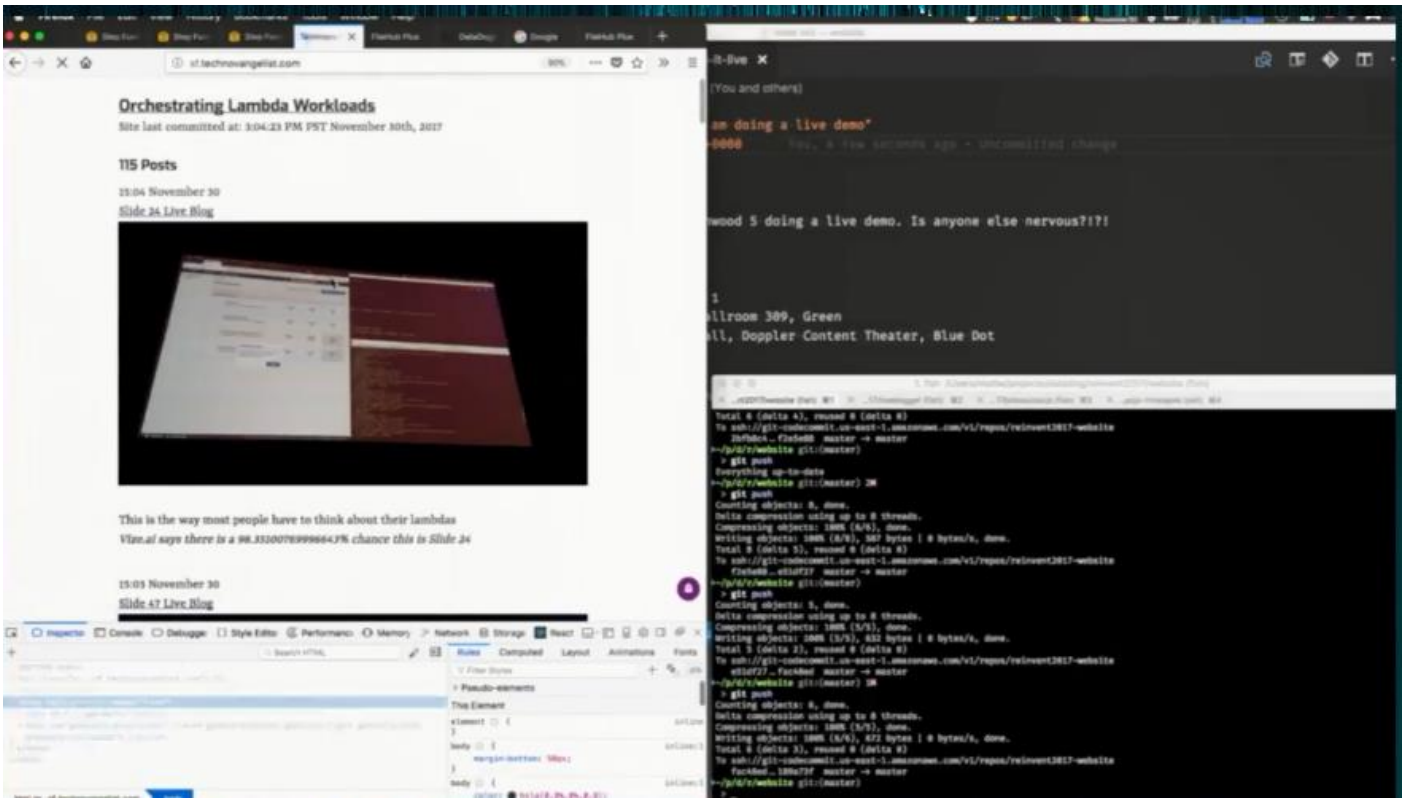


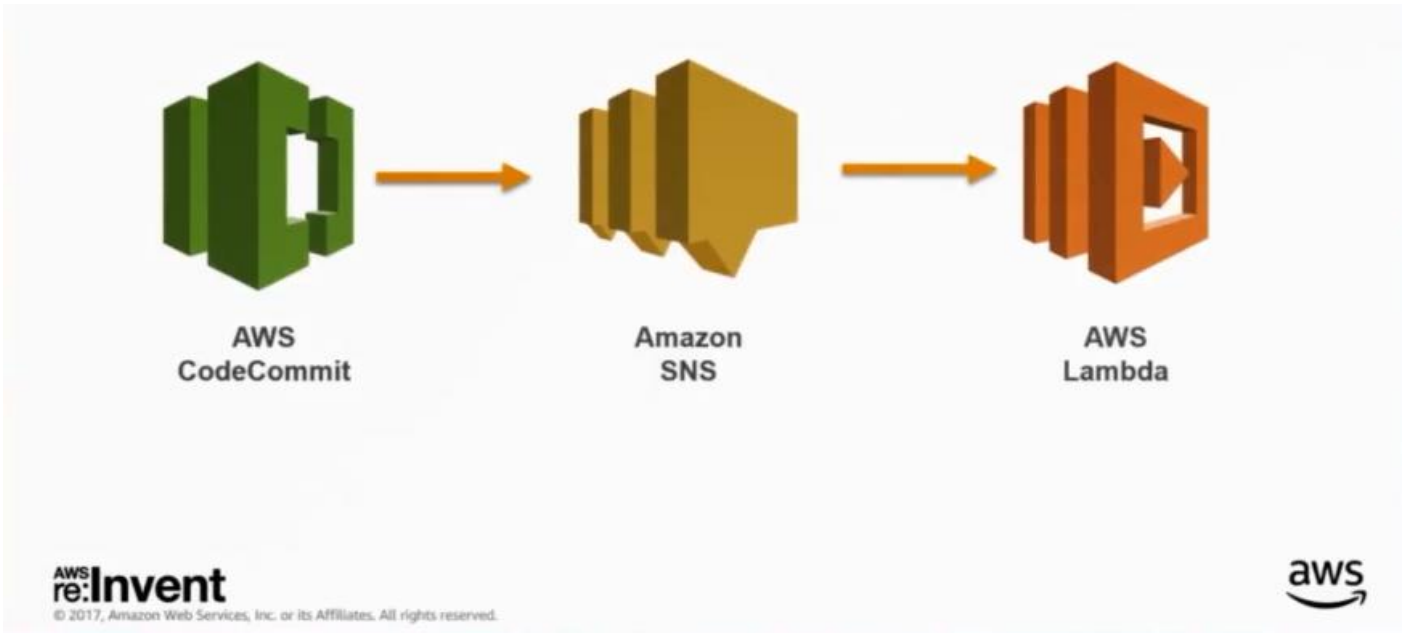


Here is the state machine that is running the whole site building and deployment from conference pictures right now


Start
WaitaSecond
WhatChanged
BuildDockerOrNo
Build
WaitForI
wood 5 doing a live demo. Is anyone else nervous?!?!
llroom 309, Green
ll, Doppler Content Theater, Blue Dot


Dashboard
Tasks
2f49048e-e7cf-417f-aec4-1d40f55cdc41
Execution Details
ContainerBuil
BuildSite
WaitForBuild
IsItBuilt
BuiltOrNot
Step Details
wood 5 doing a live demo. Is anyone else nervous?!?!
llroom 309, Green
ll, Doppler Content Theater, Blue Dot





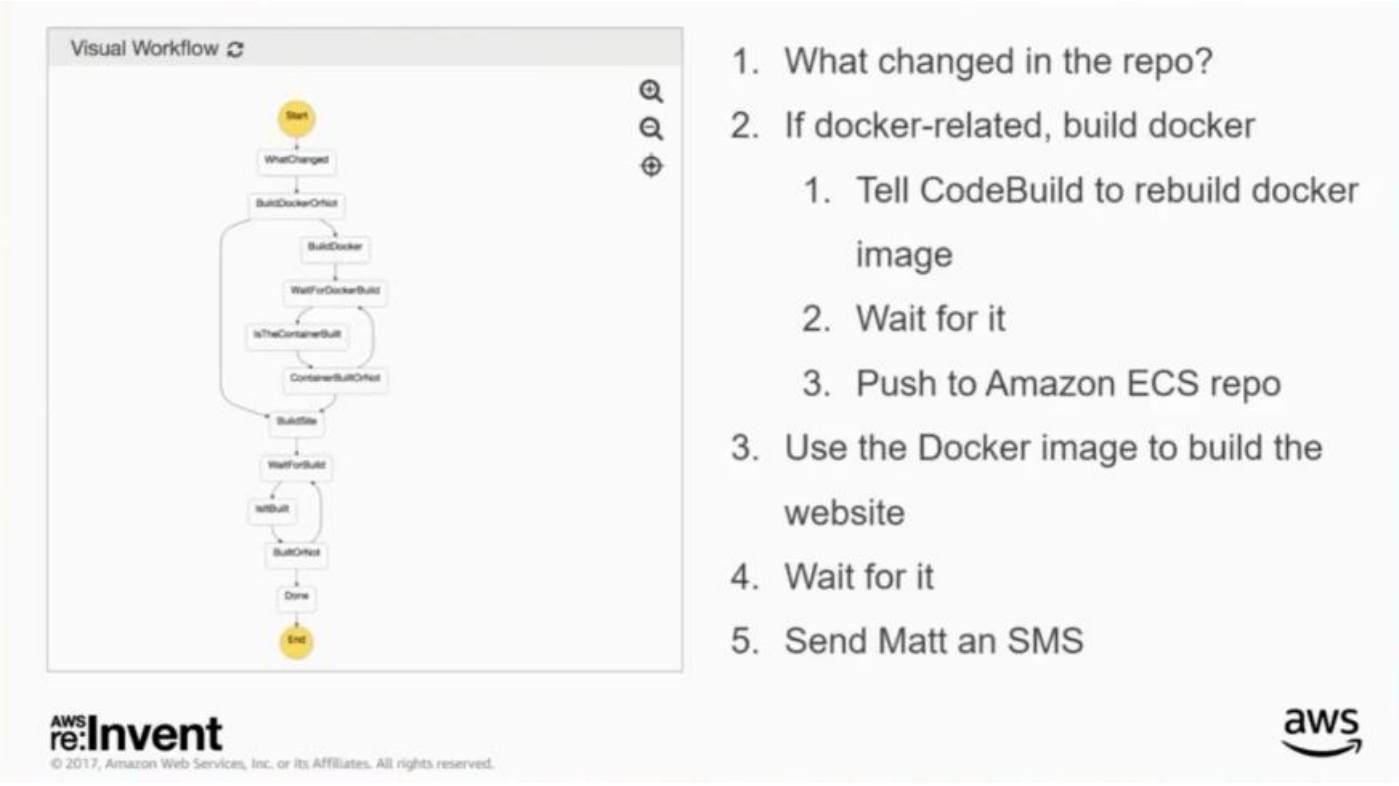
When we committed code into ***CodeCommit*** it triggered off an event to an ***SNS*** topic, we already have a ***lambda*** waiting for that SNS topic to run

```
startStepFunction:
  handler: handler.startStepFunction
  events:
    - sns: arn:aws:sns…mattw-reinvent-websitebuild
  environment:
    statemachine_arn: ${self:reso…puts.MyStateMachine.Value}
```

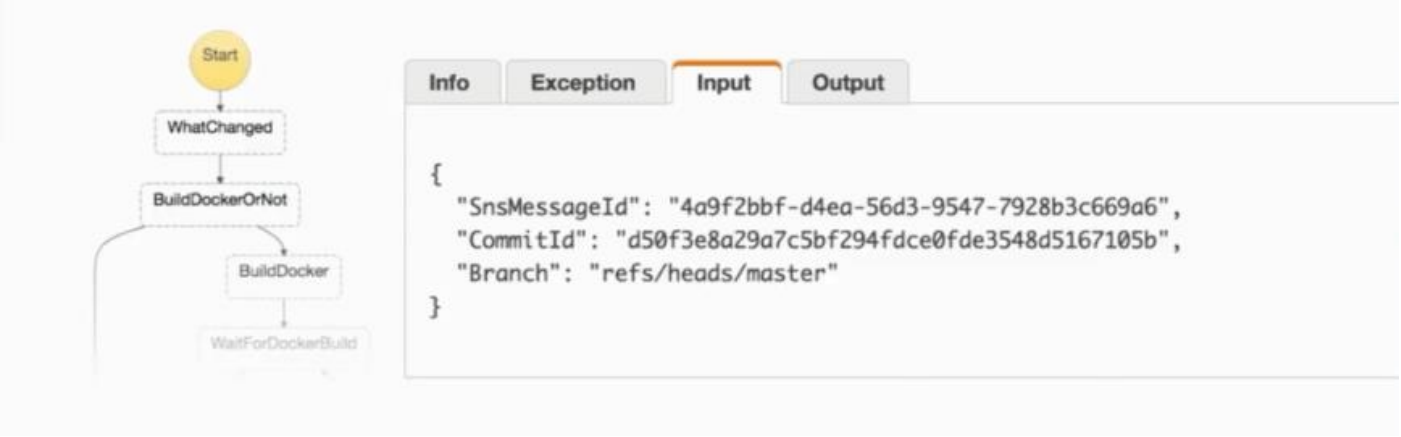
This is our definition for this ***startStepFunction*** step in our serverless.yml function. The handler called handler.sstarttepFunction is waiting for an SNS topic, we have also given it an environment variable

StartStepFunction
- Get Step Function ARN
- Get SNS details
- Start SF Execution

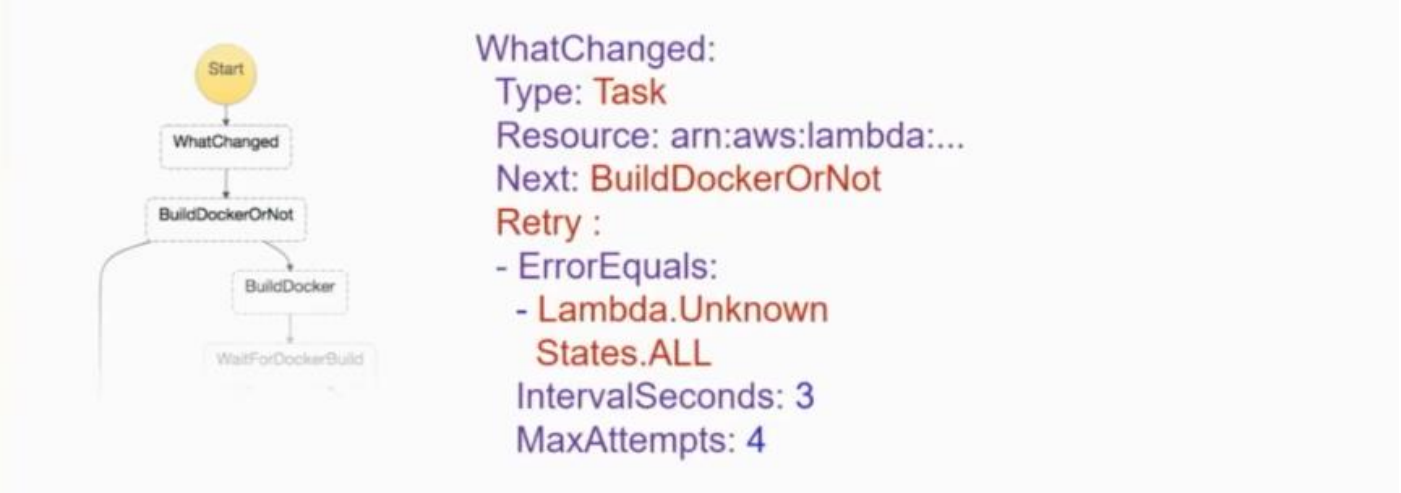
The ***startStepFunction*** basically gets the step function ***arn*** from ***serverless***, then it gets the SNS details, and then starts a step function execution.



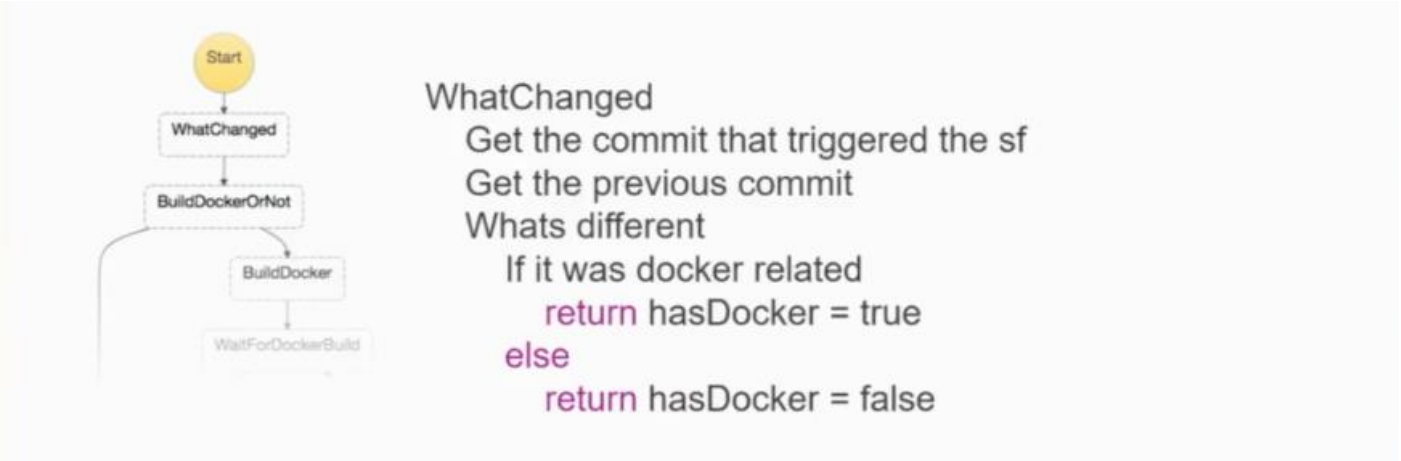
This is the step function SF that gets started. The SF is asking what and doing the steps above. It will ask what changed, if it's a file related to Docker then it will build a new image using CodeBuild.



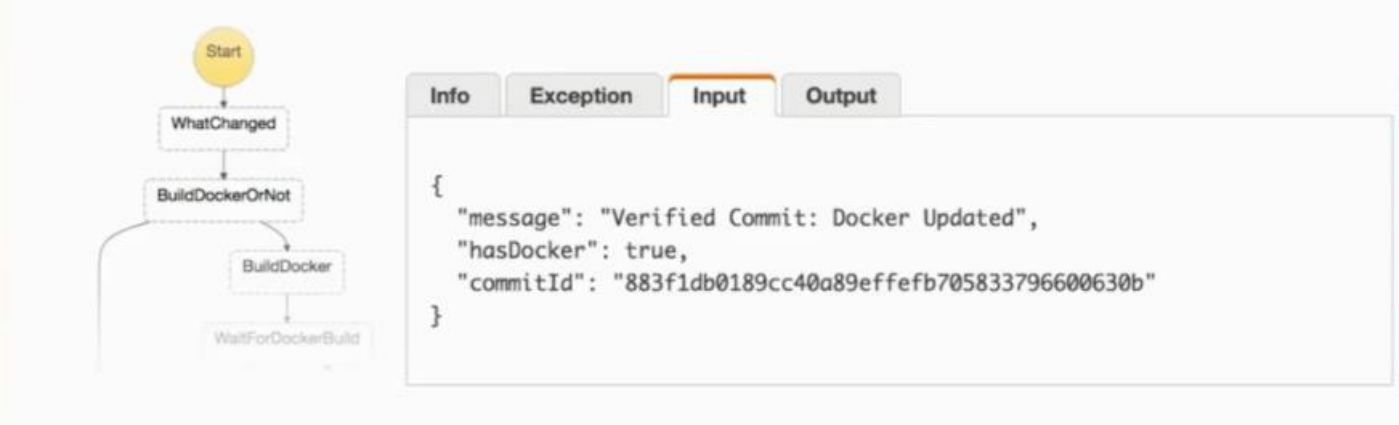
This is the type of interface that you use when working with step functions, you can do some debugging and see information related to any step/task/state. The Input tab shows you what JSON data is going into that particular task, and the Output tab shows you the output produced from that particular task.

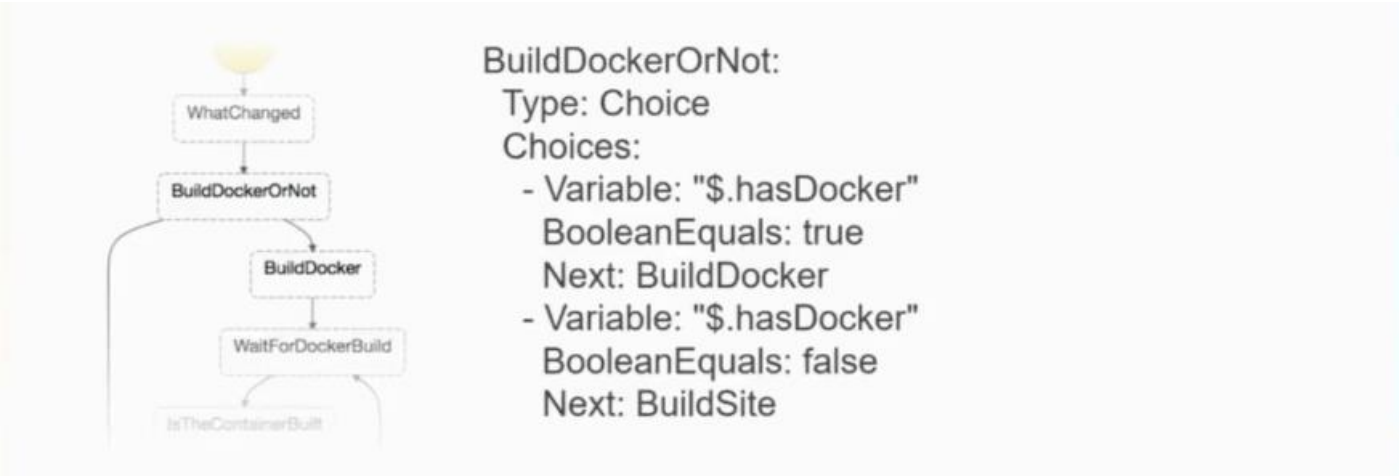


This is the ***whatChanged*** task in the ***serverless.yml*** file, we also have 4 retries



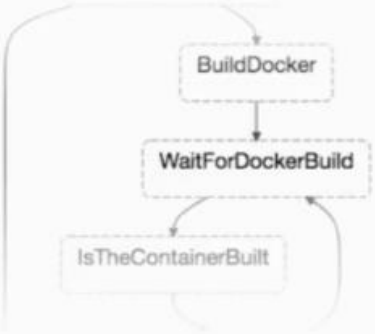
This is the ***pseudocode*** for the ***whatChanged*** task



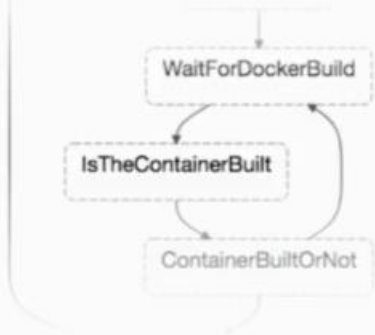


Note that we needed to add the needed IAM role permissions for this in the ***serverless.yml*** file. This will trigger a ***CodeBuild*** step as below

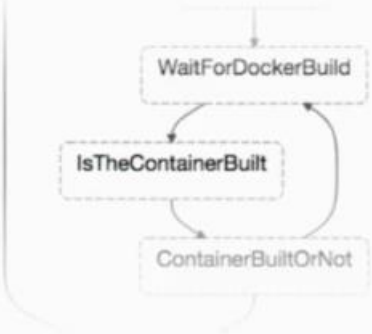
```
phases:
  install:
    commands:
      - nohup /usr/local/bin/dockerd -G dockremap --host=unix:///…
      - timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - $(aws ecr get-login --no-include-email --region us-east-1)
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t mattw-stepfunction-demo:latest .
      - docker tag mattw-stepfunction-demo:latest ...dkr.ecr.us-east-1.amazona...
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push ...dkr.ecr.us-east-1.amazonaws.com/mattw-stepfunc...
```


```
WaitForDockerBuild:
  Type: Wait
  Seconds: 20
  Next: IsTheContainerBuilt
```

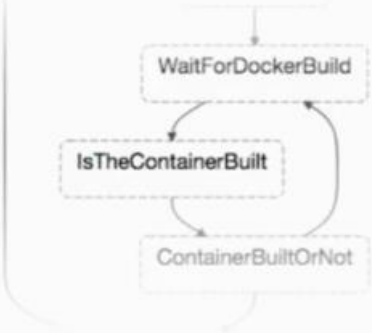



```
IsTheContainerBuilt:
  Type: Task
  Resource: arn:aws:lambda...
  Next: ContainerBuiltOrNot
```

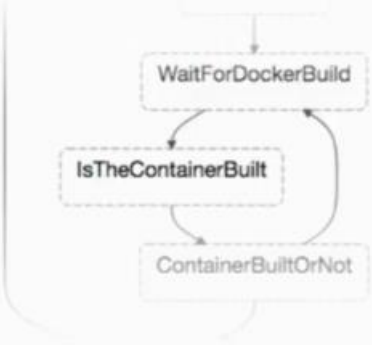


Info | Exception | Input | Output


```
{
  "message": "The container has not been built",
  "containerBuildId": "buildDockerImage:9f9600d1-a028-406d-bb6f-7053f693
  "containerBuildDone": false,
  "commitId": "883f1db0189cc40a89effefb705833796600630b"
}
```

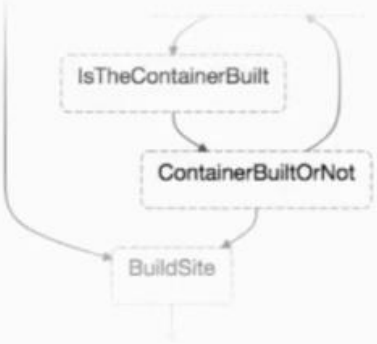



IsTheContainerBuilt
Get build info from CodeBuild
Check if its done



Info | Exception | Input | Output

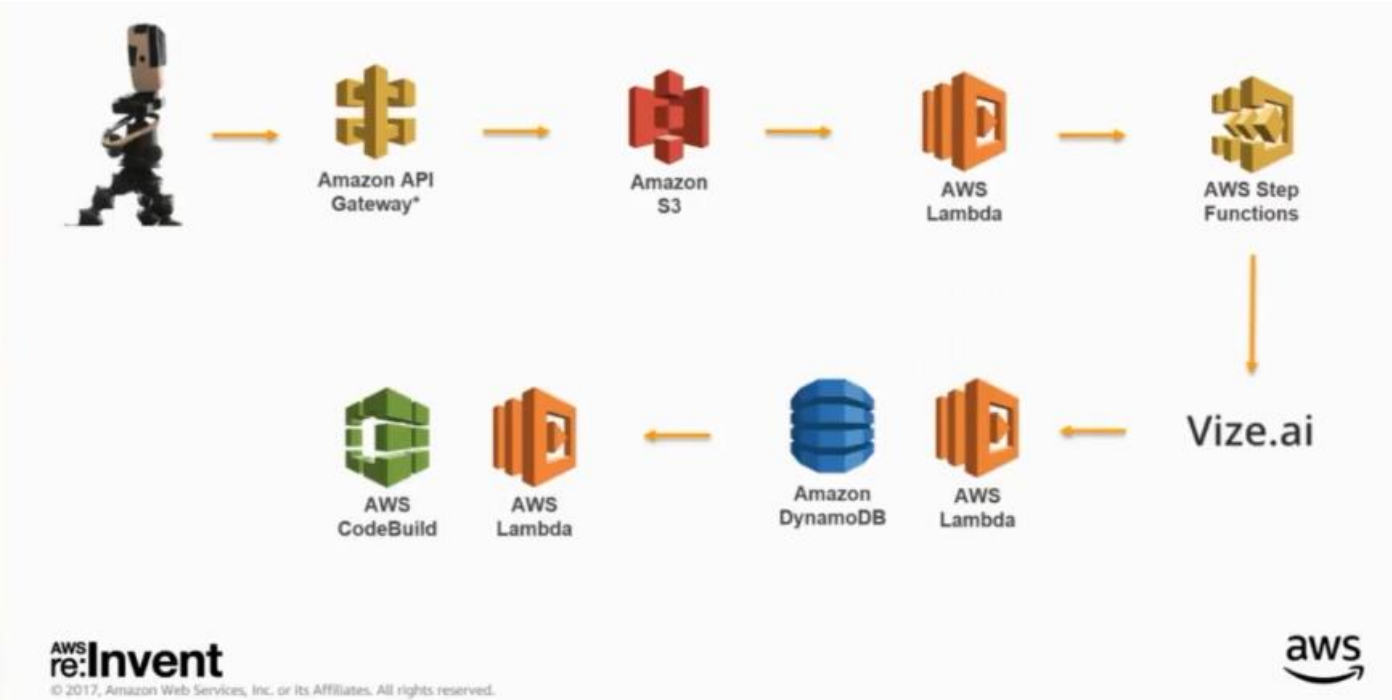

```
{
  "message": "The container has been built",
  "containerBuildId": "buildDockerImage:9f9600d1-a028-406d-bb6f-7053f693
  "containerBuildDone": true,
  "commitId": "883f1db0189cc40a89effefb705833796600630b"
}
```

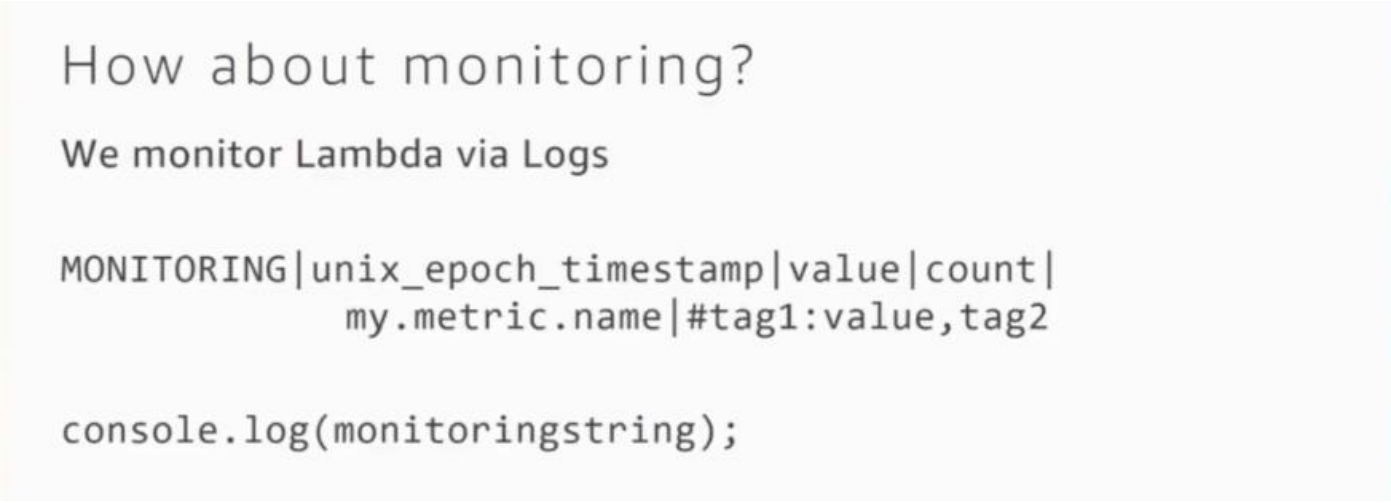

```
ContainerBuiltOrNot:
  Type: Choice
  Choices:
    - Variable: "$.containerBuildDone"
      BooleanEquals: true
      Next: BuildSite
    - Variable: "$.containerBuildDone"
      BooleanEquals: false
      Next: WaitForDockerBuild
```

```
phases:
  build:
    commands:
      - echo "****************** In  Build ******************"
      - aws s3 sync s3://mattw-reinvent2017-rawsitepages src/posts
      - export GATSBY_BUILDTIME=$(tail -1 .git/logs/HEAD | awk '{print $5}')
      - ln -s /backupmod/node_modules ./node_modules
      - gatsby build
      - echo "************** Build Complete ****************"
  post_build:
    commands:
      - echo "*************** In Post Build *****************"
      - aws s3 sync public s3://mybucket --delete --acl public-read
      - echo "************ Post Build Complete **************"
      # - command
artifacts:
  files:
    - public/**/*
```

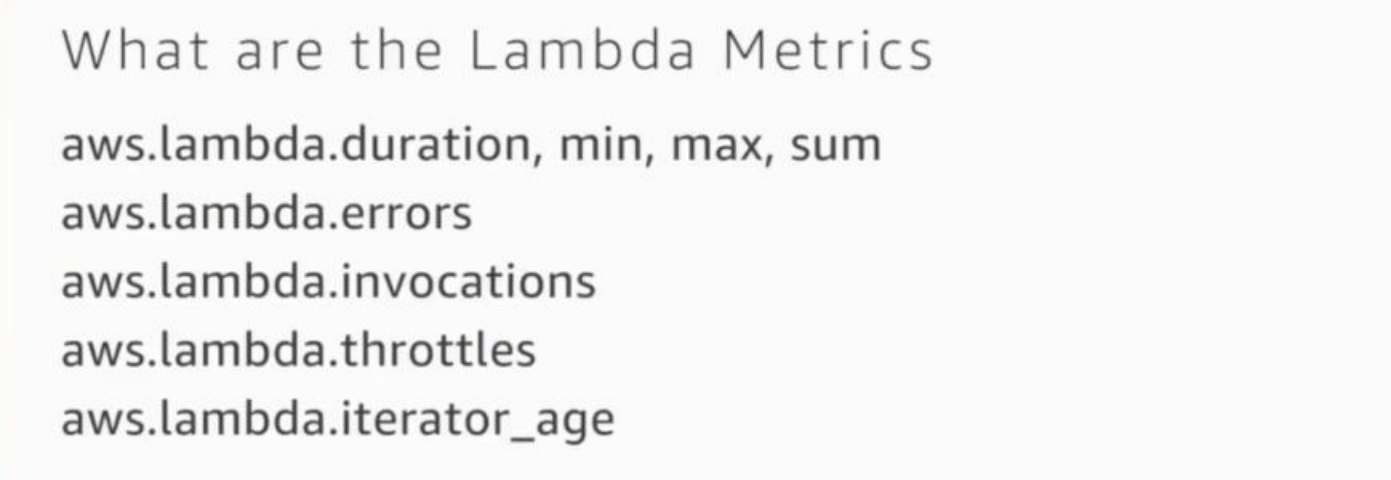

The ***buildspec.yml*** file for the ***Gatsby*** site above is a little different because we don't need to build a ***Docker*** image. We are generating the ***markdown*** files and then building the Gatsby site before doing a sync back to the ***S3*** bucket hosting our site



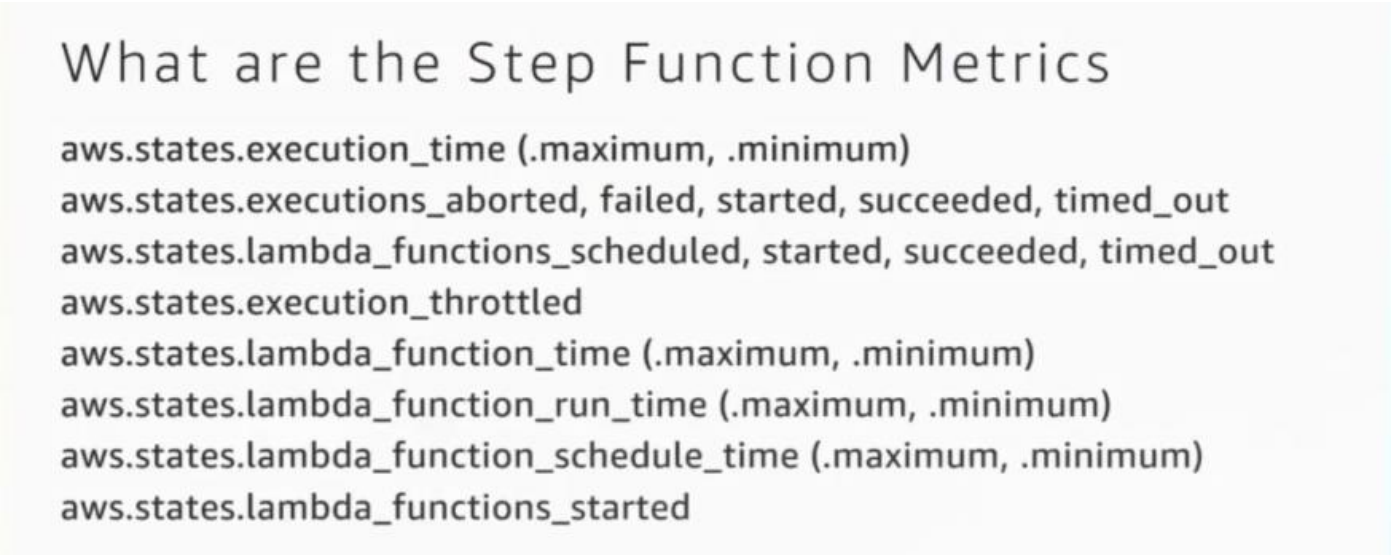
Here is a view of the workflow we have been seeing in ***Step Functions***, we start by taking pictures using a ***RaspberryPi***. We went to ***Vize.ai*** service website and uploaded about ***6000 images*** having many different rotations of the slides that I am going to be presenting at the conference today and trained the network to identify what slide number is presented to it with very high accuracy. We are then using this service to take real time image of the presentation slide at the conference and sending it to the Vize.ai service endpoint to get what slide we are on and get the slide text from ***DynamoDB*** to post on our website for live blog updates. We can also use the ***Amazon Rekognition*** service instead of Vize.ai.



We do monitoring in 2 ways, using ***CloudWatch metrics*** where we are collecting a lot of data about the different lambdas execution times, how many failures are occurring, etc. we are also collecting similar metrics from the step functions. But we can also do custom metrics, this is done inside your lambda function where you simply create and write a line to your log file as above.



Here are the metrics that we grab from CloudWatch every 15 secs,



And here are the Step Function metrics that we also collect every 15 secs and kept for a 15 months duration.

Summary

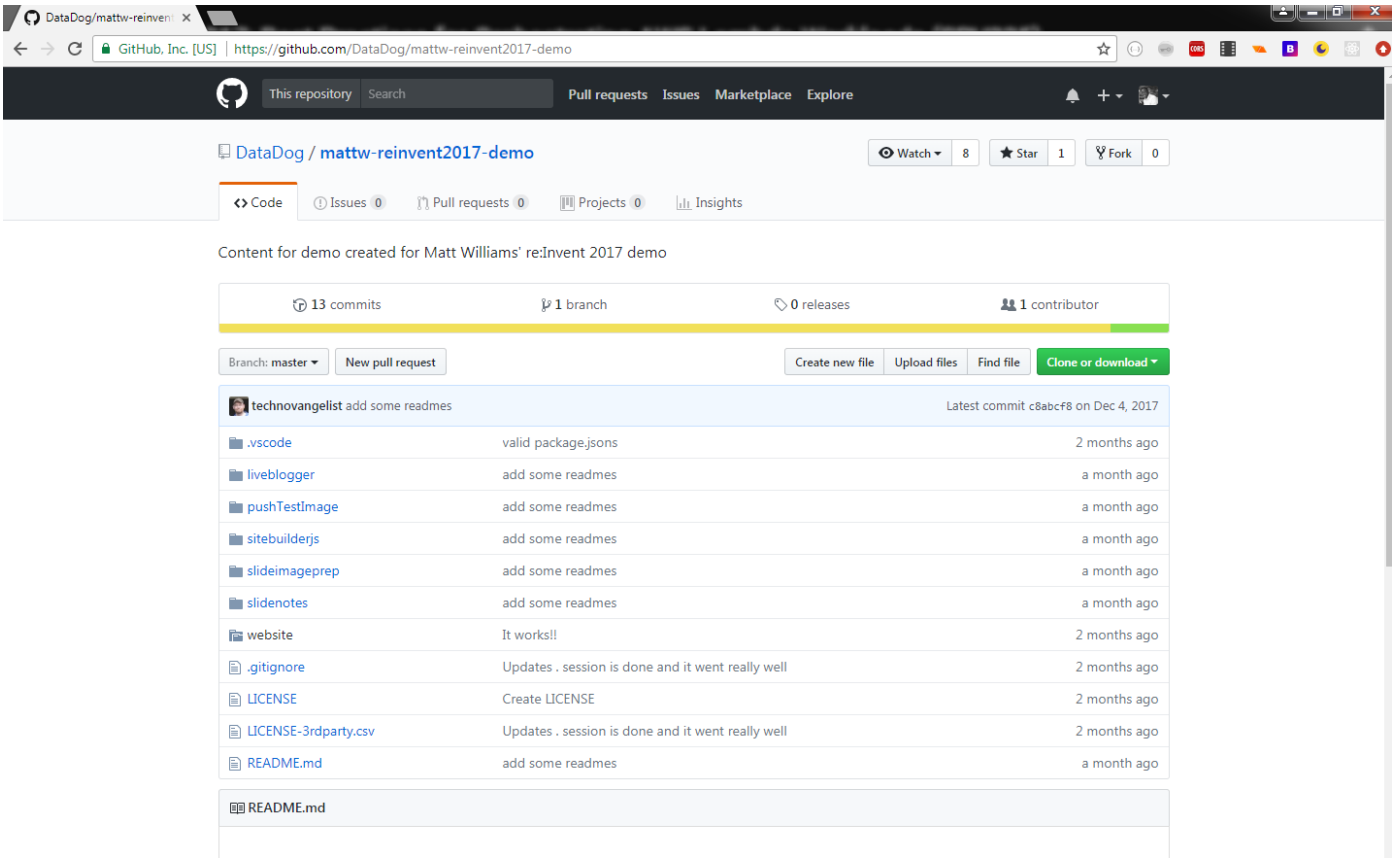
Step Functions make AWS Lambda orchestration easy

Lambda and Step Functions can be an interesting part of your infrastructure

Monitor your Lambda functions to ensure peak performance

Source can be found at:

https://github.com/DataDog/mattw-reinvent2017-demo




AWS re:Invent
THANK YOU!
Matthew Williams - @technovangelist - Datadog
AWS re:Invent
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
aws