AWS re:Invent

ARC374-P

Powering Zillow 3D Home Tours with AWS Serverless

Nikhil Gupta
Software Engineer
Zillow

Randy Yao
Software Engineer
Zillow

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws

Agenda

1. Introduction to Zillow 3D Home
2. Our Computational Problem
3. Early Prototypes
4. Exploring Alternatives
5. Going Serverless
6. Benefits
7. Next Steps

Zillow

Introduction

- Zillow is the largest real estate marketplace in the U.S.
  - “To build the largest, most trusted, and vibrant home-related marketplace in the world”
  - Empower users with data, knowledge, and inspiration in all things home
- Buyers want a way to get a feel for a home before visiting it
- Photos are limited in their authenticity
- Videos are still largely curated and don’t put the power directly into the shoppers’ hands

Zillow®

Zillow 3D Home

- Tool that allows buyers to experience a first-person walkthrough of a home before visiting it

- Allows for quick triage so shoppers can spend time visiting the homes they most care about

- Easy to capture, quick to process, and fun to explore

Zillow®

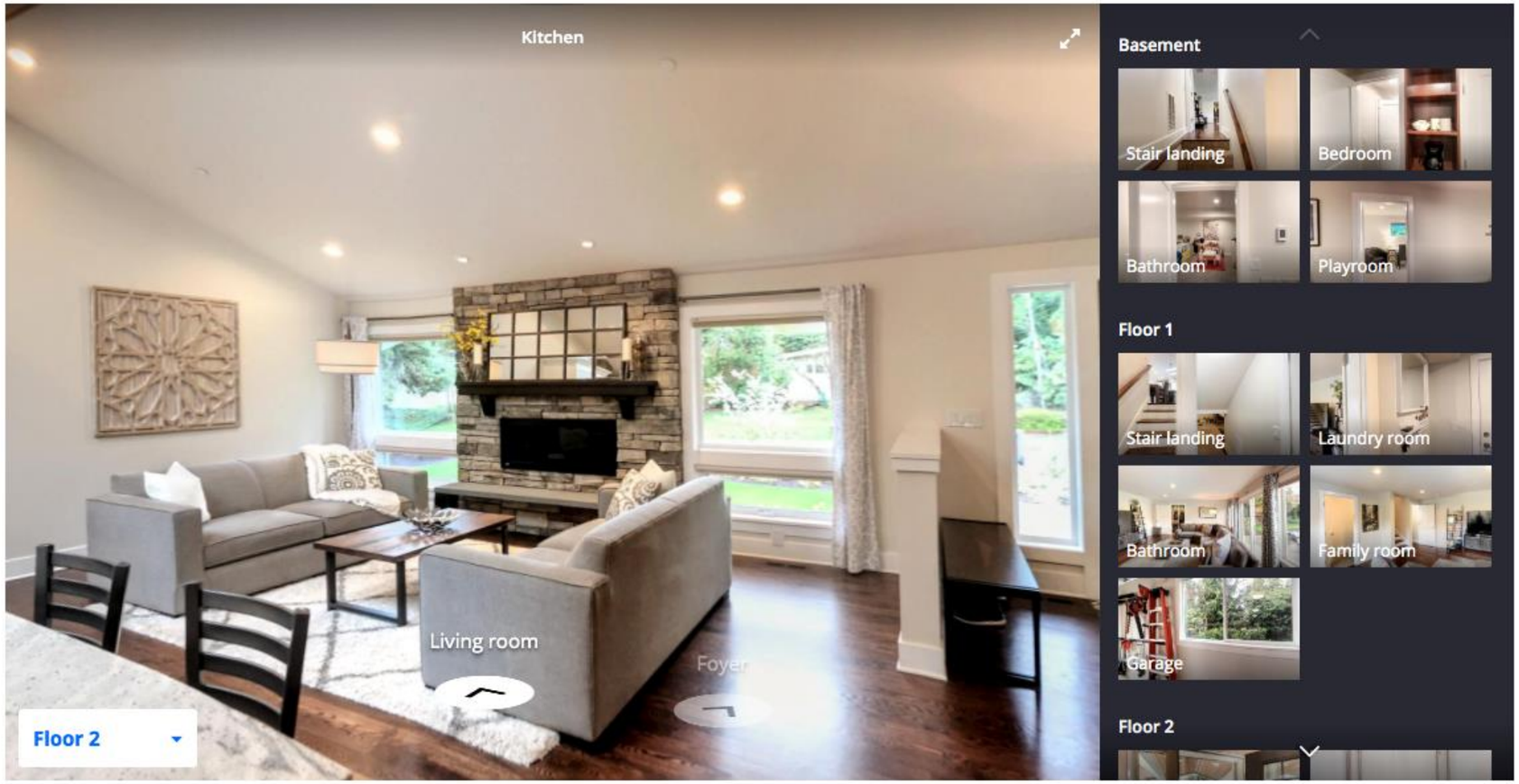
The Goal

Consumer:

- Authenticity
- Understanding
- Control

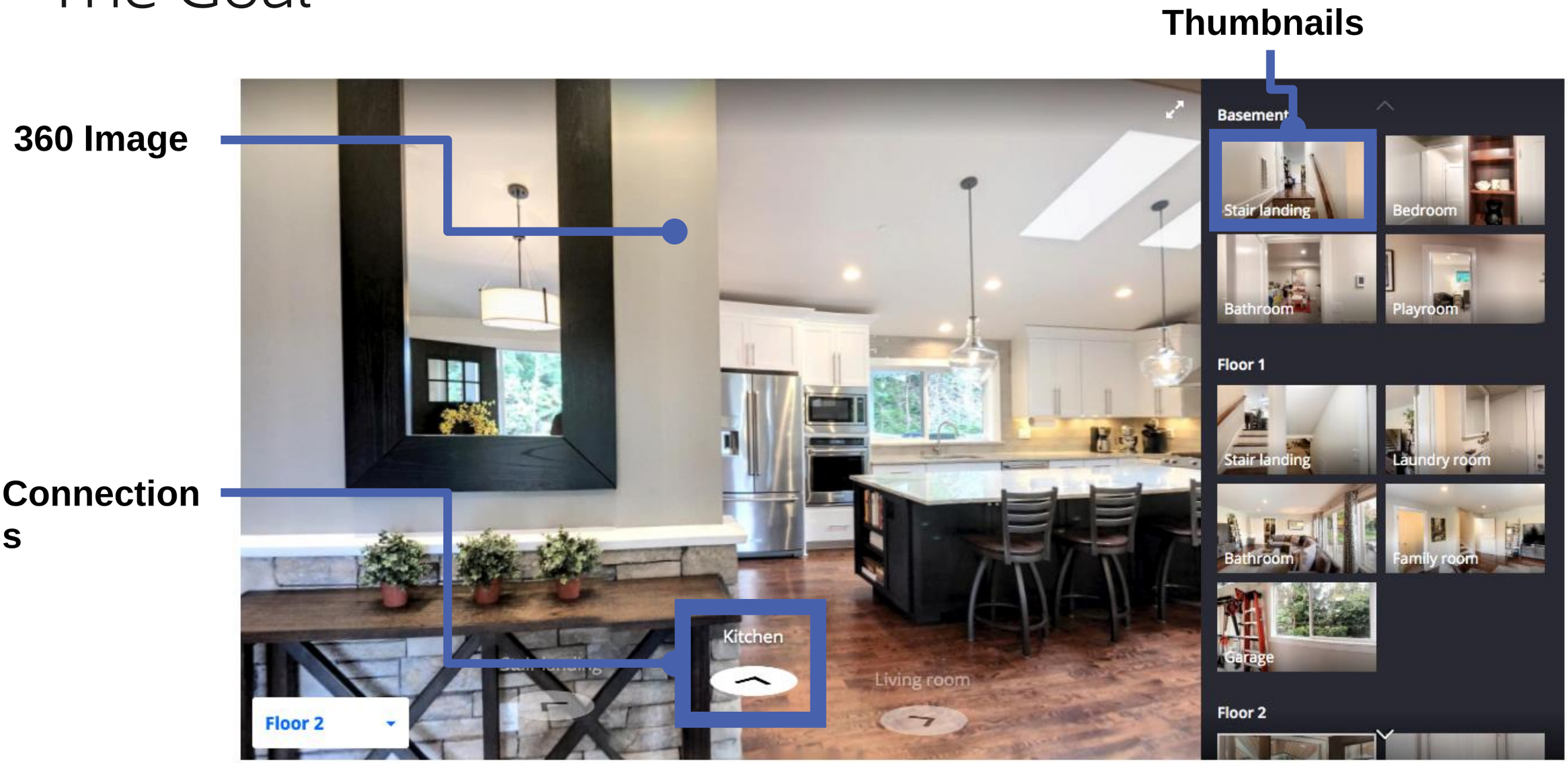
Creator:

- Free
- Easy
- Engaging



Zillow

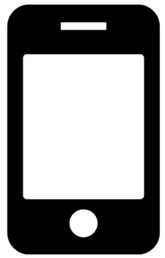
The Goal


Thumbnails
360 Image
Connections
Basement
Stair landing
Bedroom
Bathroom
Playroom
Floor 1
Stair landing
Laundry room
Bathroom
Family room
Garage
Floor 2
Kitchen
Living room
Floor 2



Data and Processing Requirements

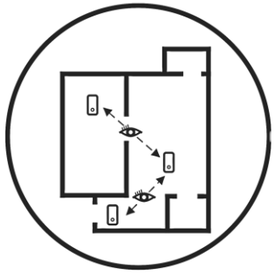
- Capture 360 images from a mobile app and upload them to our data store
- Generate preview images, thumbnails, and other image assets
- Detect visual connections between adjacent panoramas



Use 3D Home App on iOS



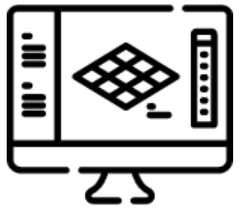
Capture 360° Images



Detect Visual Connections



Process 360° Images

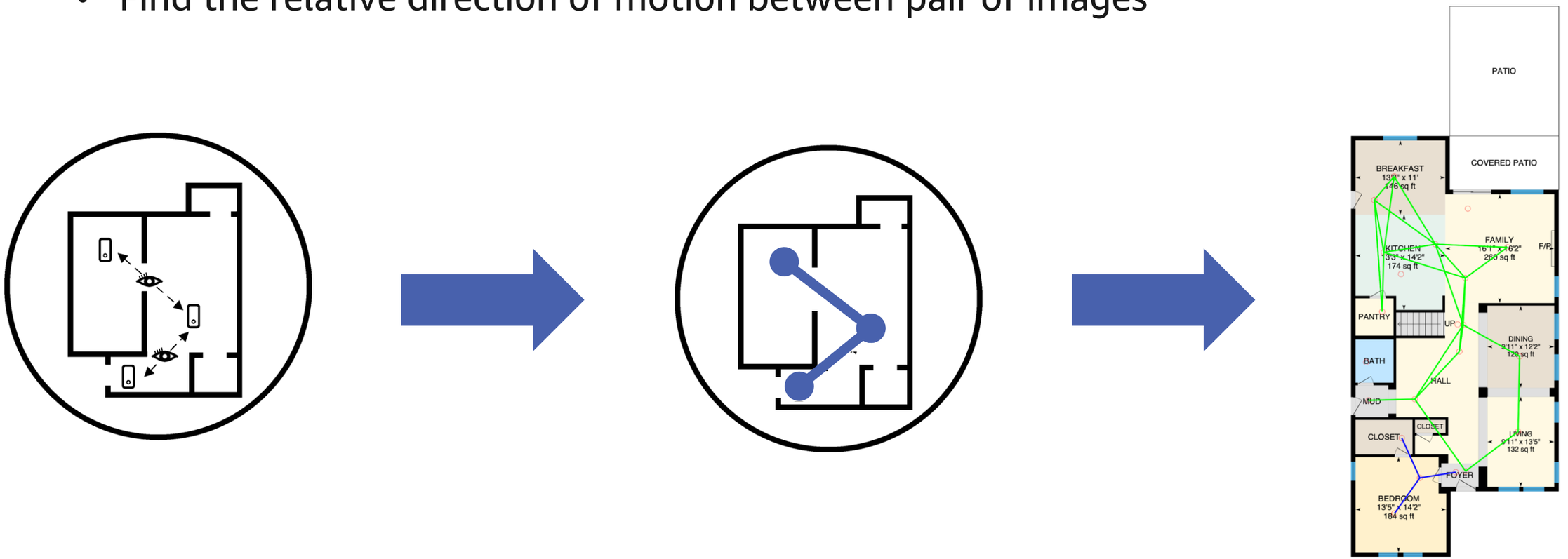


Display on Zillow.com

Zillow

Connections

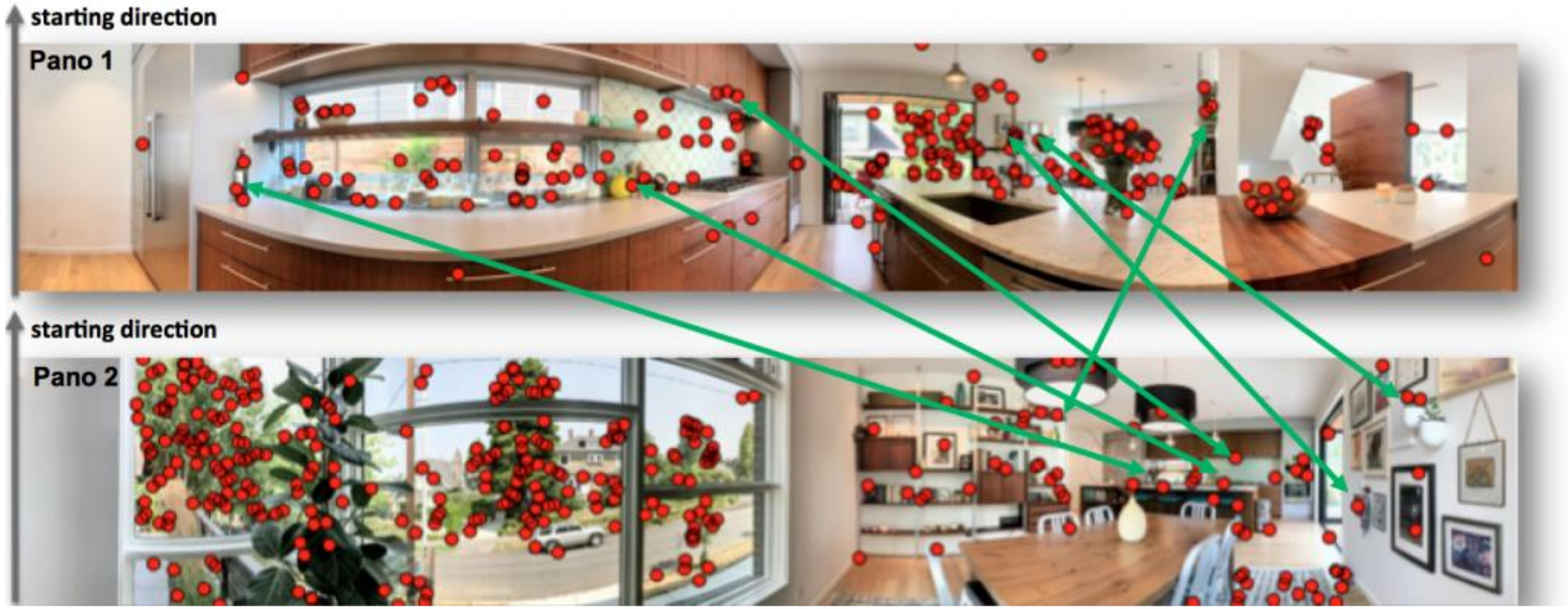
- Connections between panoramas?
  - Using computer vision
  - Find the relative direction of motion between pair of images



Zillow

Connections

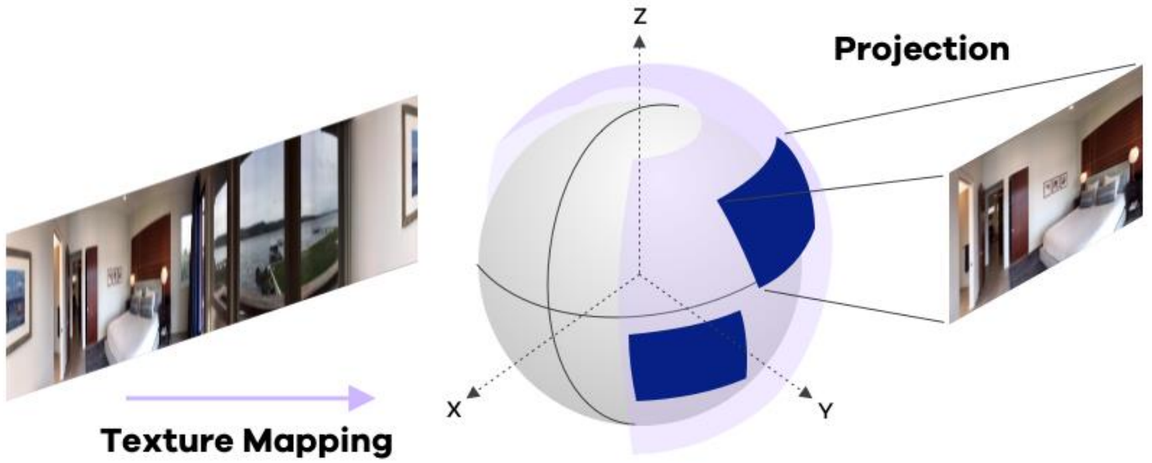
- Feature Matching
  - Find salient points
  - Describe them
  - Find the nearest neighbor in feature space



Zillow

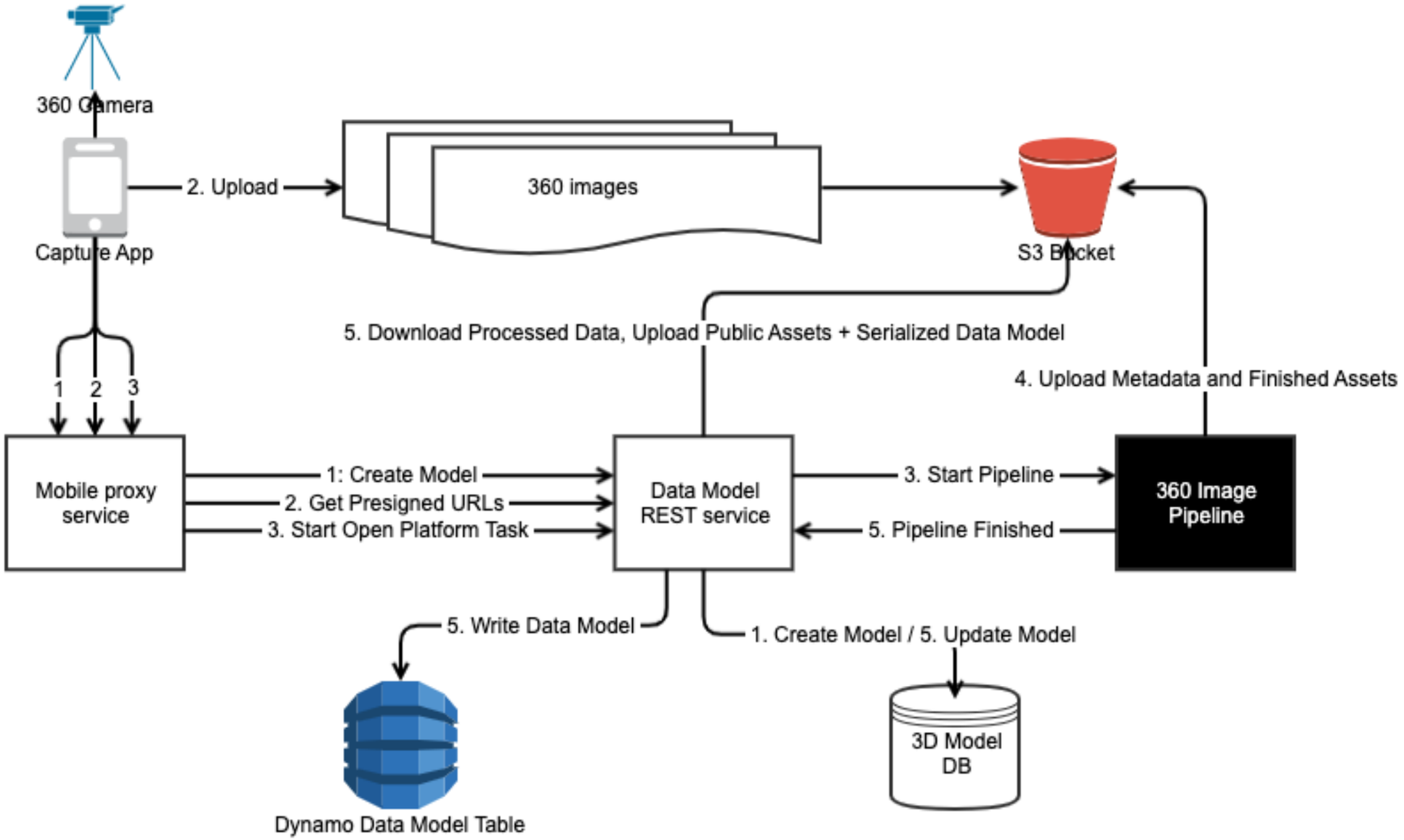
Panorama Thumbnails

- How to extract a thumbnail from a panorama
  - Using computer vision
  - Get planar projection
- Think of it like “unwrapping a sphere”



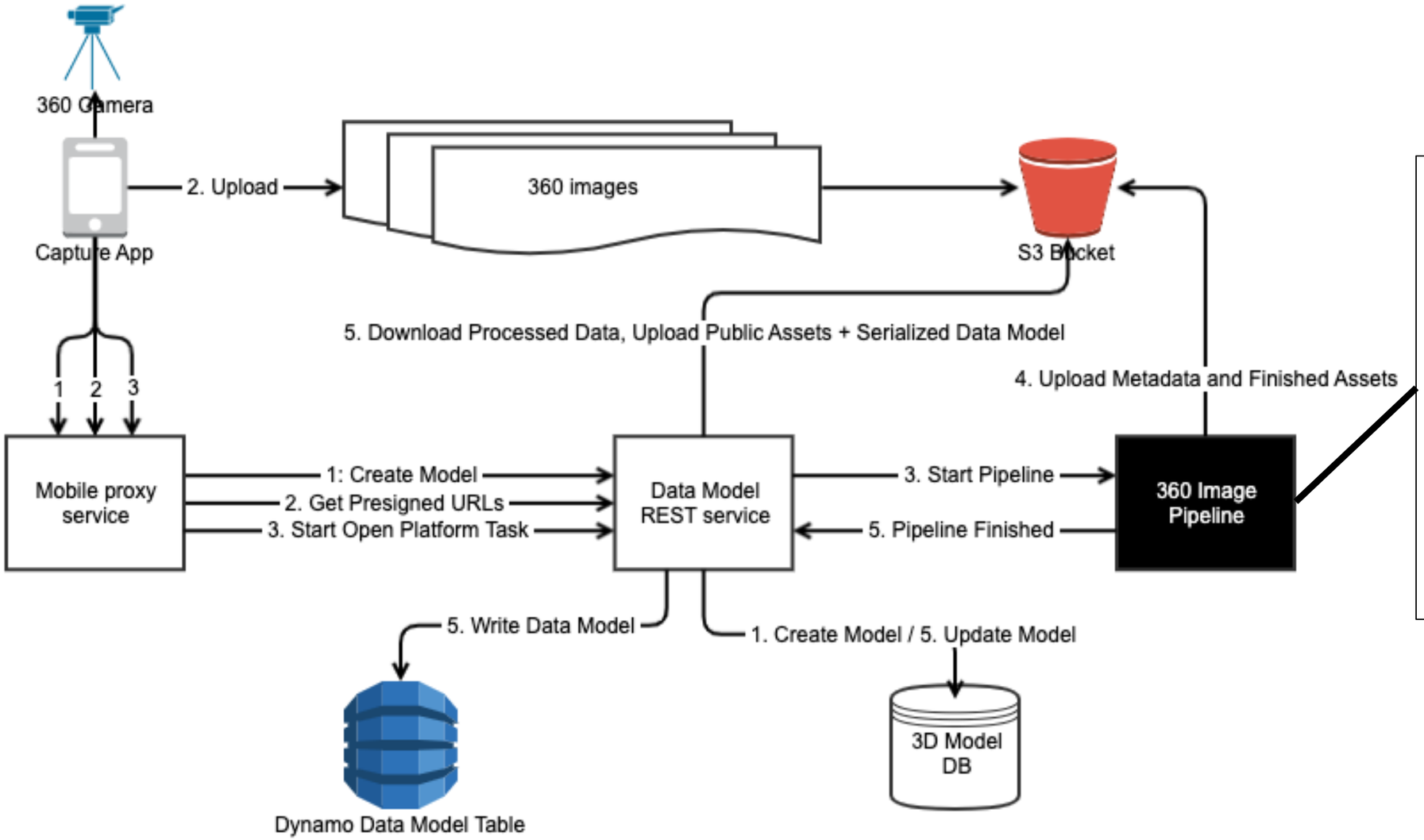
Zillow

High-Level Architecture



Zillow

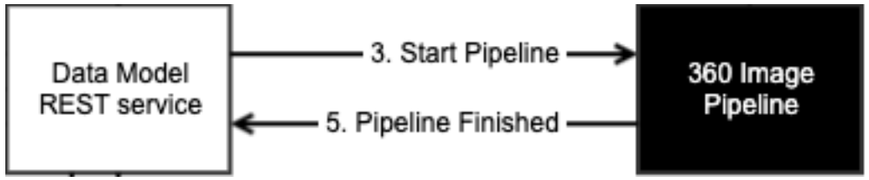
High-Level Architecture



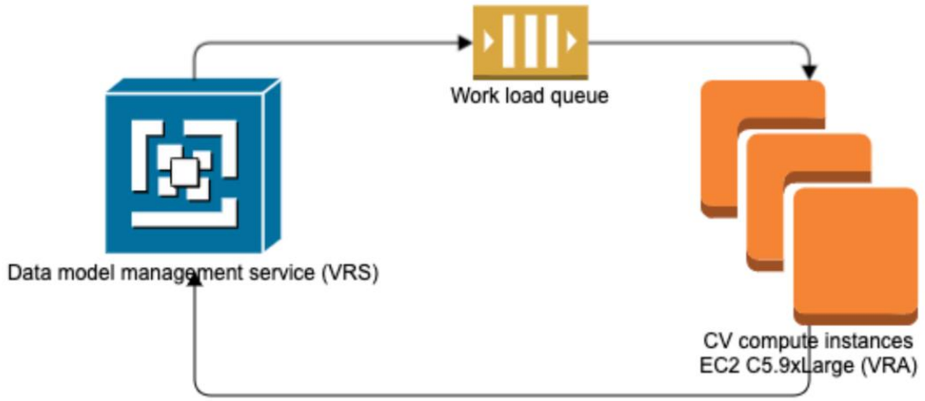
- Vision computations are costly
- Thumbnail computation load scales linearly
- Connections load scales with pairs of pano images

Zillow

Prototype and Development Beta

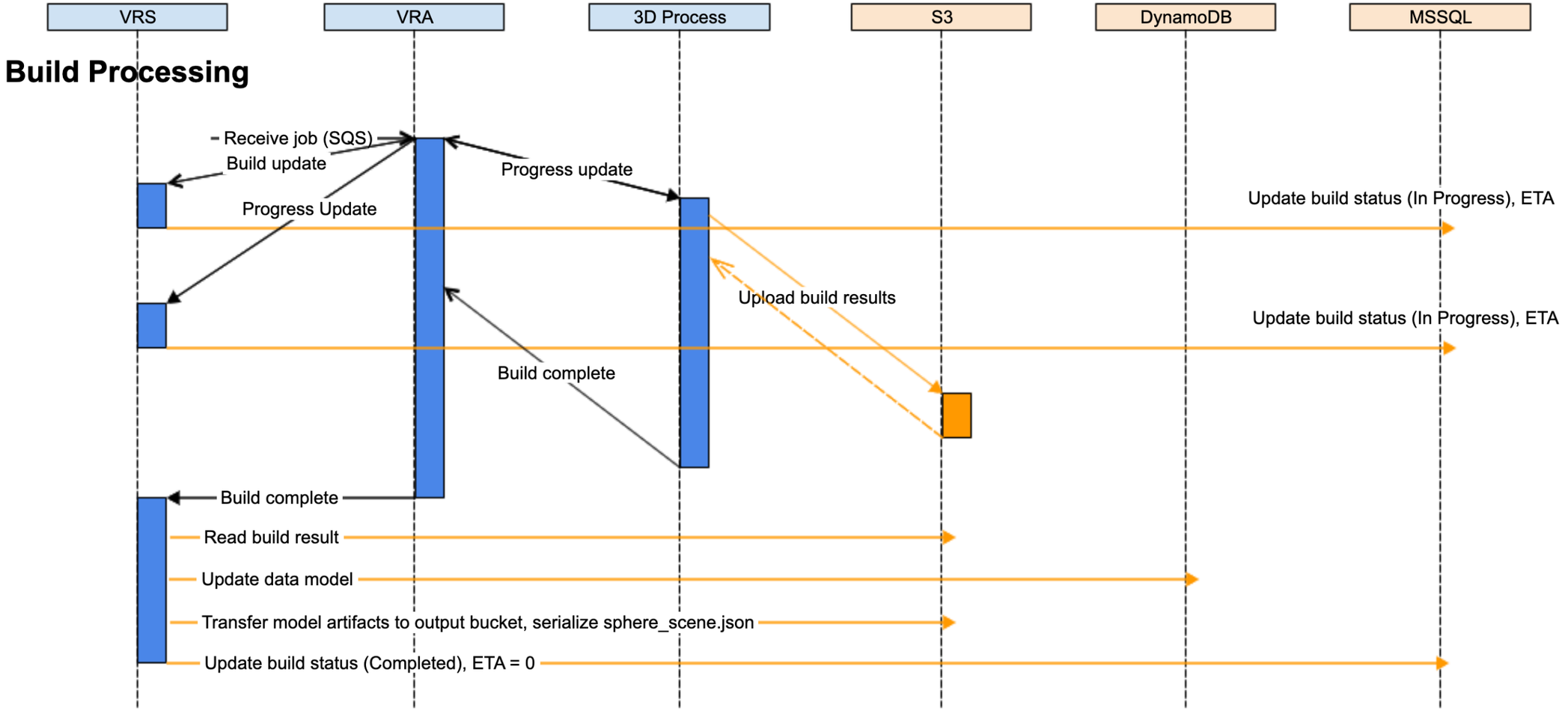






Zillow

Prototype and Development Beta



Zillow

Prototype and Development Beta

- Costly
  - Multiple AWS EC2 c5.9xlarge hosts
  - Underutilized computational resources
- Hard to maintain
  - Tracing execution status through the pipeline difficult, messy
  - Difficult to debug and retry
  - Chatty communication with data API, lots of points of failure
  - Managing host configurations and software instances
  - Managing host service state

Zillow

Back to the Drawing Board

- How do we better handle our spiky workload
- How do we get better insight into the status of a single execution
- How do we reduce cost
- How do we build this to be extensible to support future workloads

Zillow

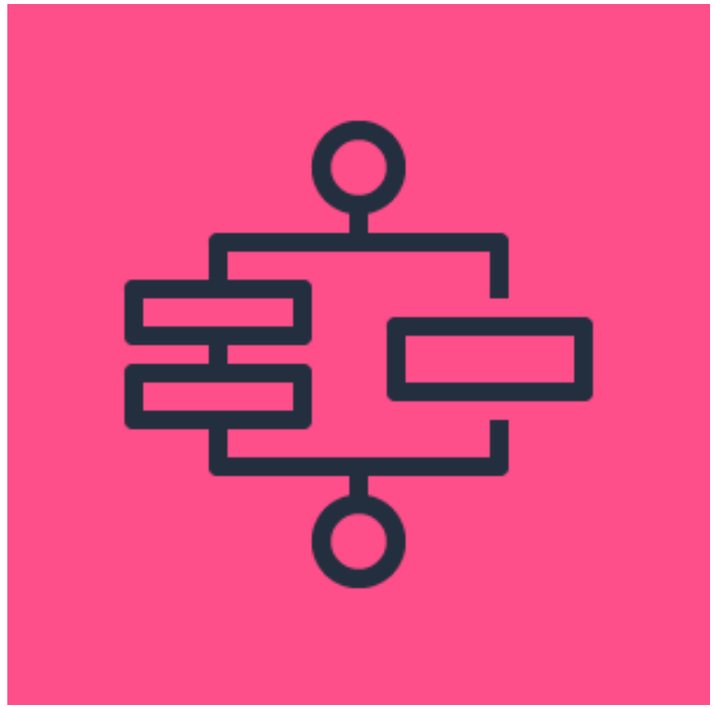
Options We Considered

- New custom workflow orchestration service
- AWS Simple Workflow
- Airflow
- Luigi
- AWS Step Functions

Zillow

Options We Considered

- New custom workflow orchestration service
- AWS Simple Workflow
- Airflow
- Luigi
- AWS Step Functions



The winner: AWS Step Functions

Zillow

Deep Dive into Step Functions

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws

What are AWS Step Functions

- They provide a workflow orchestration framework
- Step functions are distributed state machines
- Serverless
  - No managing hosts
  - Scales up on demand
  - Instantaneous availability
- Built-in integration with other AWS services
- Easily extensible design

Zillow

How is a Step Function Defined

- State machines are defined in JSON using the Amazon States Language
- Consists of individual states and the transitions between them

State machine definition

```
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass state",
  "StartAt": "Start",
  "States": {
    "Start": {
      "Type": "Pass",
      "Result": {
        "otherData": "woof",
        "vals": [1, 2, 3, 4]
      },
      "Next": "Filter"
    },
    "Filter": {
      "OutputPath": "$.vals[0]",
      "Type": "Pass",
      "End": true
    }
  }
}
```



Zillow

States

- Building blocks of your Step Function
- Output of one state becomes input of another
- Different types depending on your need
  - Pass
  - Task
  - Choice
  - Wait
  - Succeed
  - Fail
  - Parallel
  - Map (new)

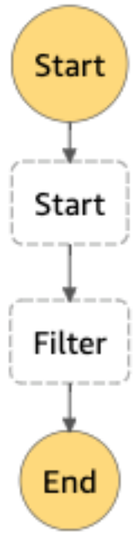
Zillow

Example:

- Start state outputs some values that become the input to the next state
- Filter state selects the first element of the *vals* array and outputs it
- $ represents the current input
- Can pass in our own data to the first state

State machine definition

```
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass state",
  "StartAt": "Start",
  "States": {
    "Start": {
      "Type": "Pass",
      "Result": {
        "otherData": "woof",
        "vals": [1, 2, 3, 4]
      },
      "Next": "Filter"
    },
    "Filter": {
      "OutputPath": "$.vals[0]",
      "Type": "Pass",
      "End": true
    }
  }
}
```



Zillow

Task States

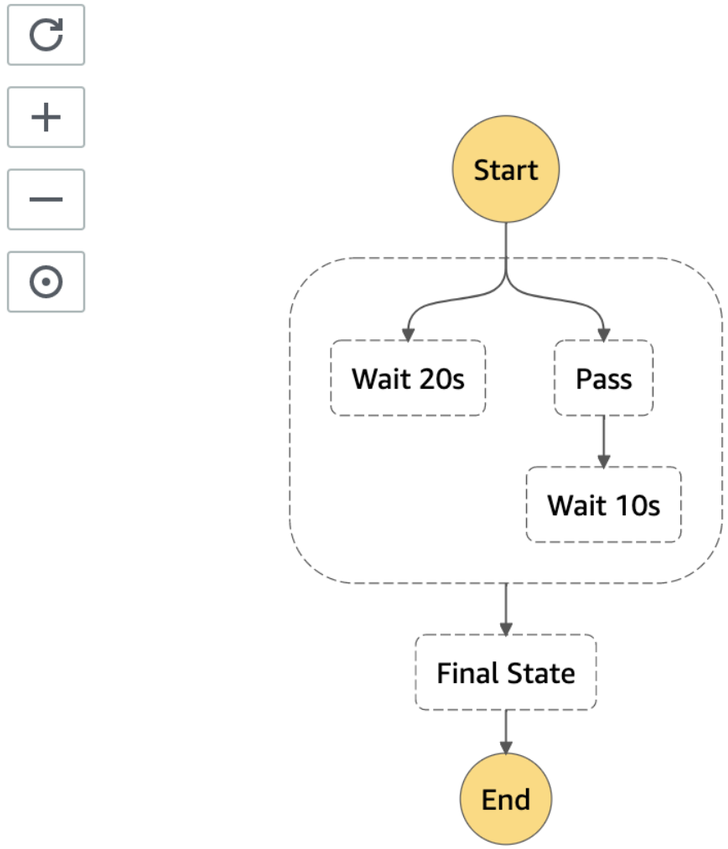
- Perform the real “work”
- Integrate directly with AWS services
  - Invoke lambdas
  - Send SNS messages
  - Perform DynamoDB operations
  - Run an ECS job

```
"LambdaState": {
  "Type": "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:HelloWorld",
  "Next": "NextState"
}
```

Zillow

Parallel Blocks

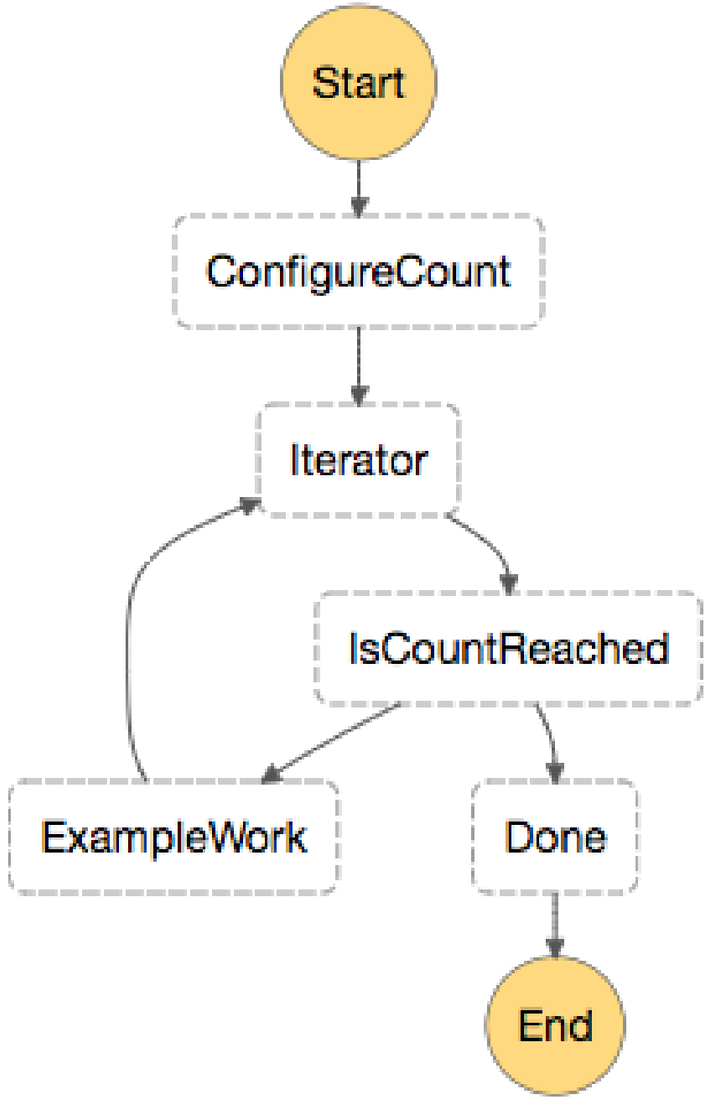
- You can create a parallel task type to execute multiple branches simultaneously
- Extremely useful for any independent tasks to decrease your total execution time
- Returns an array of each branch's results to the next state after the Parallel block



Zillow

Iterating Workloads

- If you need to loop inside a state machine, use an iterator pattern
- Iterator will grab next input, increment count, or whatever it needs to do to “move forward”
- IsCountReached is a choice state that will indicate if the loop is complete
- ExampleWork will do your actual work



Zillow

Key Workflows

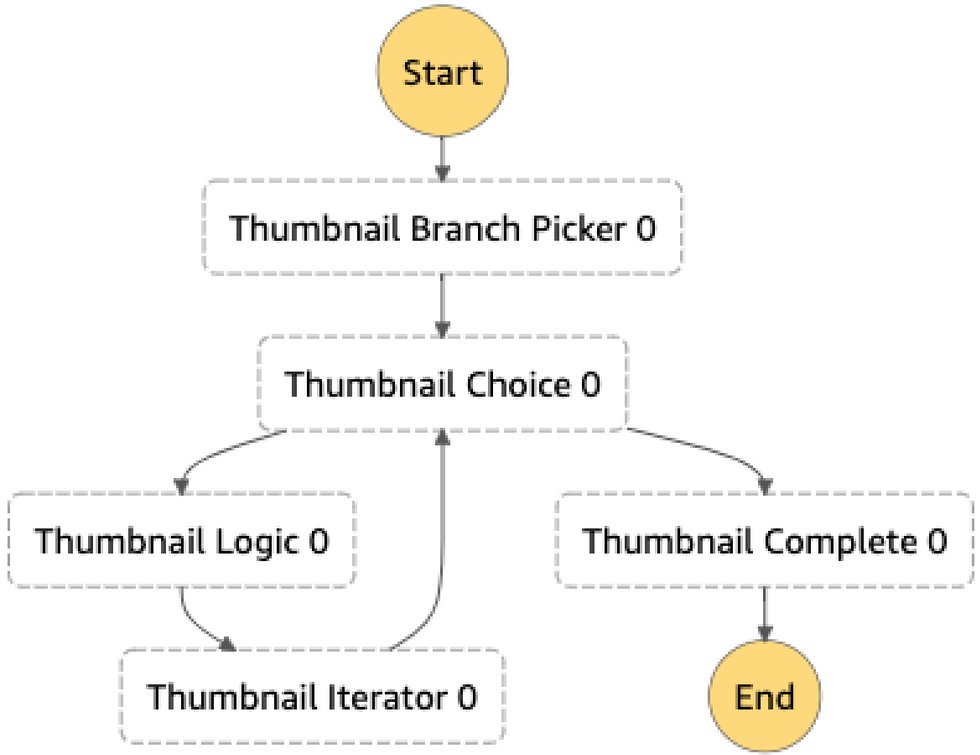
- Thumbnail and other image asset generation
  - Lower resolution preview images for app, listing, and browsing purposes
  - Tiled images for faster rendering in tour viewer
  - One execution per image

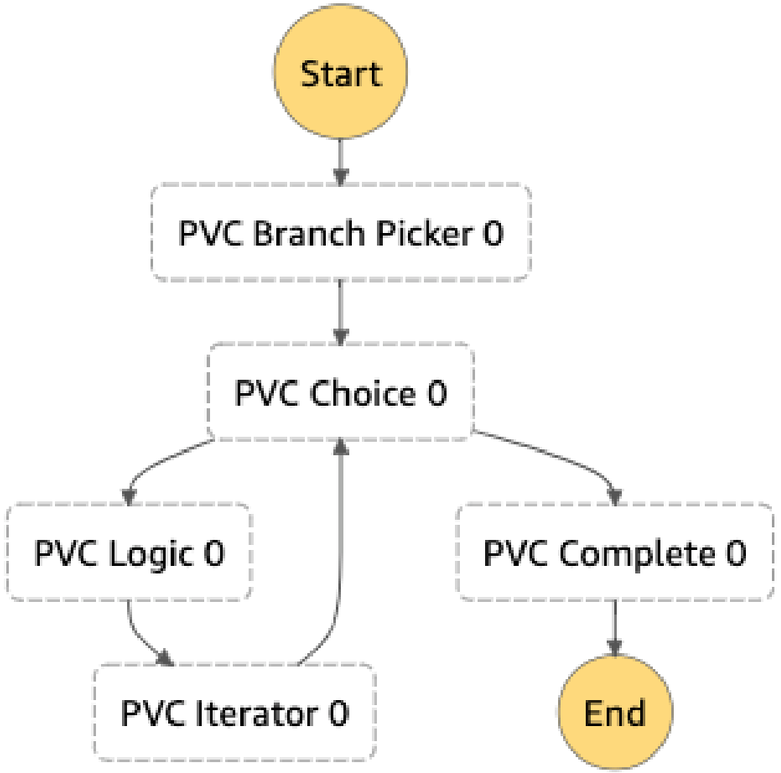
- Pano-Visual Connection detection
  - Computer vision task to determine if two panoramas are linked to each other (e.g. two rooms in the same home share an entrance/exit)
  - One execution per image pair

Both of these tasks are highly parallelizable on a per-pano or per-pair basis.

Zillow

Key Workflows





Zillow

Providing Payload

- Lambda to generate payload for a given workflow
  - Thumbnail splitter lambda
  - PVC splitter lambda

- Downloads input manifest provided by app to get the list of panoramas

- Generates a list of payloads *for its workflow*
  - Per-pano payload for thumbnail generation
  - Per-pair payload for PVC generation

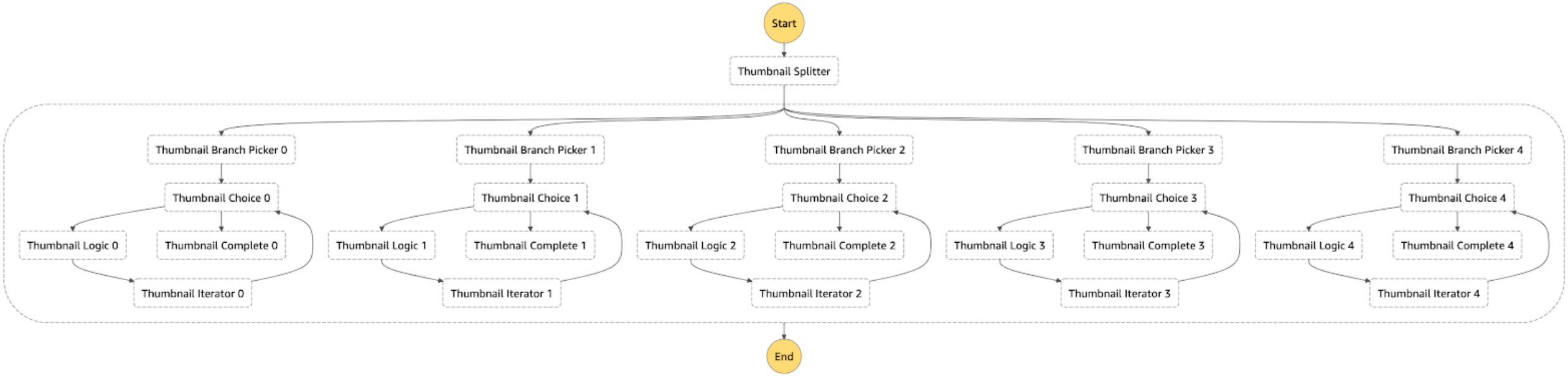
Zillow

Example: Thumbnail Splitter output


```
[{
    "vr_model_id": 101,
    "vr_model_guid": "00076b6e-c3c4-4e43-8675-b3f01f5b267e",
    "thumbnail_schema_version": "0.0.1",
    "entity_id": "abcdefabcde",
    "s3_output_prefix": "public/tours/00076b6e-c3c4-4e43-8675-b3f01f5b267e/thumbnail/abcdefabcde",
    "s3_pano_key": "raw/tours/00076b6e-c3c4-4e43-8675-b3f01f5b267e/pano/abcdefabcde/panorama.jpg",
    "start_angle": 175.0
}, {
    "vr_model_id": 101,
    "vr_model_guid": "00076b6e-c3c4-4e43-8675-b3f01f5b267e",
    "thumbnail_schema_version": "0.0.1",
    "entity_id": "bcdefabcdef",
    "s3_output_prefix": "public/tours/00076b6e-c3c4-4e43-8675-b3f01f5b267e/thumbnail/bcdefabcdef",
    "s3_pano_key": "raw/tours/00076b6e-c3c4-4e43-8675-b3f01f5b267e/pano/abcdefabcde/panorama.jpg",
    "start_angle": 175.0
},
…
]
```


Zillow

End-to-end for a Single Workflow



Zillow

Gotcha: Max Payload Size

- Step functions only allows you to transfer 32KB directly between states
- For largely parallel tasks we had issues with “infinite parallelism” because of task size
  - We hit limits at ~80 panoramas
  - Make sure to load test!
- Solved with a backing store in DynamoDB
  - Pass partition and range keys as input to states that need to use it

Zillow

Dynamo for Task Status Tracking

- Each Splitter now creates an item in DynamoDB and populates it with a set of inputs that must be processed

- Iterators now pull an input from that item, and also put their output into that same item.

Zillow

Dynamo for Task Status Tracking

```
{
   "execution_id": "5e684023-d0ca-48a3-82f9-5eb615afb229_455310",   // partition key (hash key)
   "task_type": "thumbnail",                                        // sort key (range key)
   "output": {
      "key0": <module_output0>,
      "key1": <module_output1>,
   },
   "input": {
      "key2": {
         "branch_name": null,
         "input": <module_input4>
      }
      "key3": {
         "branch_name": “branch0”,
         "input": <module_input5>
      },
      "key4": {
         "branch_name": “branch1”,
         "input": <module_input5>
      },
      ...
   },
}
```

Zillow

Handling Failures with DynamoDB Solution

- Iterator needs to be idempotent
- Failure can occur at any stage
- Make sure two iterators don’t pick up the same task
  - Conditional PutItem
  - Choose random element from remaining input

Zillow

Put Output in DynamoDB

Dynamo Item:

```
{
   "execution_id": "5e684023-d0ca-48a3-82f9-5eb615afb229_455310",
   "task_type": "thumbnail",
   "output": {
      "key0": <module_output0>,
      "key1": <module_output1>,
      “key3”: <module_output2>
   },
   "input": {
      "key2": {
         "branch_name": null,
         "input": <module_input2>
      }
      "key3": {
         "branch_name": “branch0”,
         "input": <module_input3>
      },
      "key4": {
         "branch_name": “branch1”,
         "input": <module_input5>
      },
   }
}
```

Iterator Input:

```
{
   "is_complete": false,
   "dynamo_io_key": {
      "execution_id": "5e684023-d0ca-48a3-82f9-5eb615afb229_455310",
      "task_type": "thumbnail",
   },
   "branch_name": "branch0",
   "current_task_key": "key3",
   "module_input": <module_input3>,
   "module_output": <module_output2>
}
```

Zillow®

Delete Current Input

Dynamo Item:

```
{
   "execution_id": "5e684023-d0ca-48a3-82f9-5eb615afb229_455310",
   "task_type": "thumbnail",
   "output": {
      "key0": <module_output0>,
      "key1": <module_output1>,
      “key3”: <module_output2>
   },
   "input": {
      "key2": {
         "branch_name": null,
         "input": <module_input2>
      }
      "key3": {
         "branch_name": “branch0”,
         "input": <module_input3>
      },
      "key4": {
         "branch_name": “branch1”,
         "input": <module_input5>
      },
   }
}
```

Iterator Input:

```
{
   "is_complete": false,
   "dynamo_io_key": {
      "execution_id": "5e684023-d0ca-48a3-82f9-5eb615afb229_455310",
      "task_type": "thumbnail",
   },
   "branch_name": "branch0",
   "current_task_key": "key3",
   "module_input": <module_input3>,
   "module_output": <module_output2>
}
```

Zillow

Grab New *Unclaimed* Input

Dynamo Item:

```
{
   "execution_id": "5e684023-d0ca-48a3-82f9-5eb615afb229_455310",
   "task_type": "thumbnail",
   "output": {
      "key0": <module_output0>,
      "key1": <module_output1>,
      “key3”: <module_output2>
   },
   "input": {
      "key2": {
         "branch_name": null,
         "input": <module_input2>
      }
      "key3": {
         "branch_name": “branch0”,
         "input": <module_input3>
      },
      "key4": {
         "branch_name": “branch1”,
         "input": <module_input5>
      },
   }
}
```

Iterator Output:

```
{
   "is_complete": false,
   "dynamo_io_key": {
      "execution_id": "5e684023-d0ca-48a3-82f9-5eb615afb229_455310",
      "task_type": "thumbnail",
   },
   "branch_name": "branch0",
   "current_task_key": "key3",
   "module_input": <module_input3>,
   "module_output": null
}
```

Zillow

Potential for Clashing

Dynamo Item:

```
{
   "execution_id": "5e684023-d0ca-48a3-82f9-5eb615afb229_455310",
   "task_type": "thumbnail",
   "output": {
      "key0": <module_output0>,
      "key1": <module_output1>,
      “key3”: <module_output3>,
      “key4”: <module_output4>
   },
   "input": {
      "key2": {
         "branch_name": null,
         "input": <module_input2>
      }
   }
}
```

Iterator 1:

```
{
   ...,
   "branch_name": "branch0",
   "current_task_key": "key3",
  }
```

Iterator 2:

```
{
   ...,
   "branch_name": "branch1",
   "current_task_key": "key4",
}
```

Zillow

All in a Single Operation (Conditional UpdateItem)

```
dynamo_client.update_item(
    Key={"execution_id": event.dynamo_io_key.execution_id, "task_type": event.dynamo_io_key.task_type},
    UpdateExpression=(
        "SET #output.#current_task_key = :module_output, ”
        "#input.#new_task_key.#branch_name = :branch_name, "
        "REMOVE #input.#current_task_key"
    ),
    ExpressionAttributeNames={
        "#output": "output",
        "#current_task_key": event.current_task_key,
        "#input": "input",
        "#new_task_key": new_task_key,
        "#branch_name": "branch_name"
    },
    ExpressionAttributeValues={
        ":module_output": event.module_output,
        ":branch_name": event.branch_name,
        ":null": "NULL",
    },
    ConditionExpression=(
        "attribute_exists(#input.#new_task_key) AND attribute_type(#input.#new_task_key.branch_name, :null)"
    )
)
```

Zillow

Last Iteration

Dynamo Item:

```
{
   "execution_id": "5e684023-d0ca-48a3-82f9-5eb615afb229_455310",
   "task_type": "thumbnail",
   "output": {
      "key0": <module_output0>,
      "key1": <module_output1>,
      “key3”: <module_output2>
   },
   "input": {
      "key2": {
         "branch_name": “branch1”,
         "input": <module_input2>
      }
   }
}
```

Iterator Input:

```
{
   "is_complete": false,
   "dynamo_io_key": {
      "execution_id": "5e684023-d0ca-48a3-82f9-5eb615afb229_455310",
      "task_type": "thumbnail",
   },
   "branch_name": "branch0",
   "current_task_key": "key3",
   "module_input": <module_input3>,
   "module_output": <module_output2>
}
```

Zillow®

Last Iteration

Dynamo Item:

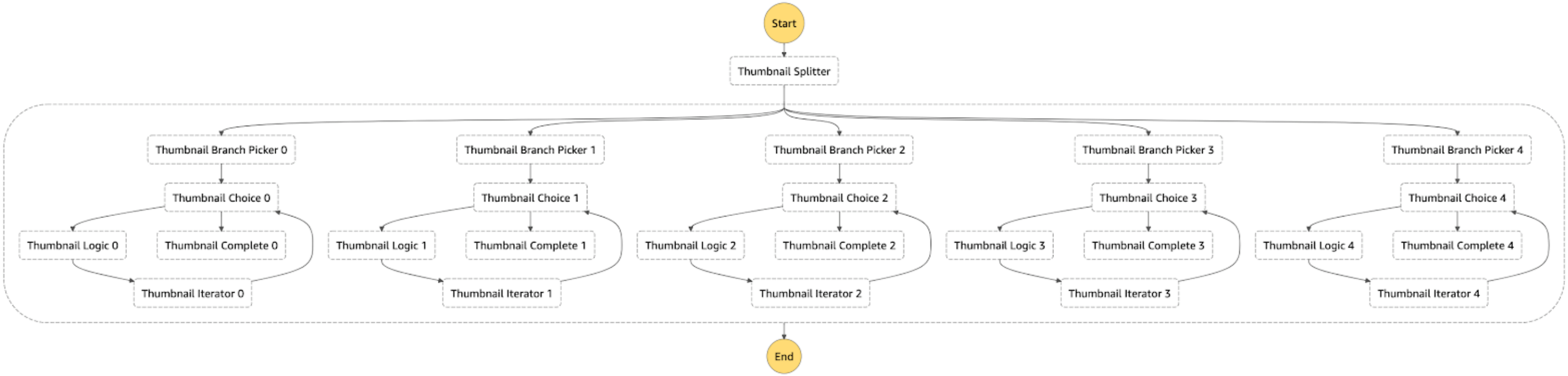
```
{
   "execution_id": "5e684023-d0ca-48a3-82f9-5eb615afb229_455310",
   "task_type": "thumbnail",
   "output": {
      "key0": <module_output0>,
      "key1": <module_output1>,
      “key3”: <module_output2>
   },
   "input": {
      "key2": {
         "branch_name": “branch1”,
         "input": <module_input2>
      }
   }
}
```

Iterator Output:

```
{
   "is_complete": true,
   "dynamo_io_key": {
      "execution_id": "5e684023-d0ca-48a3-82f9-5eb615afb229_455310",
      "task_type": "thumbnail",
   },
   "branch_name": "branch0",
   "current_task_key": null,
   "module_input": null,
   "module_output": null
}
```

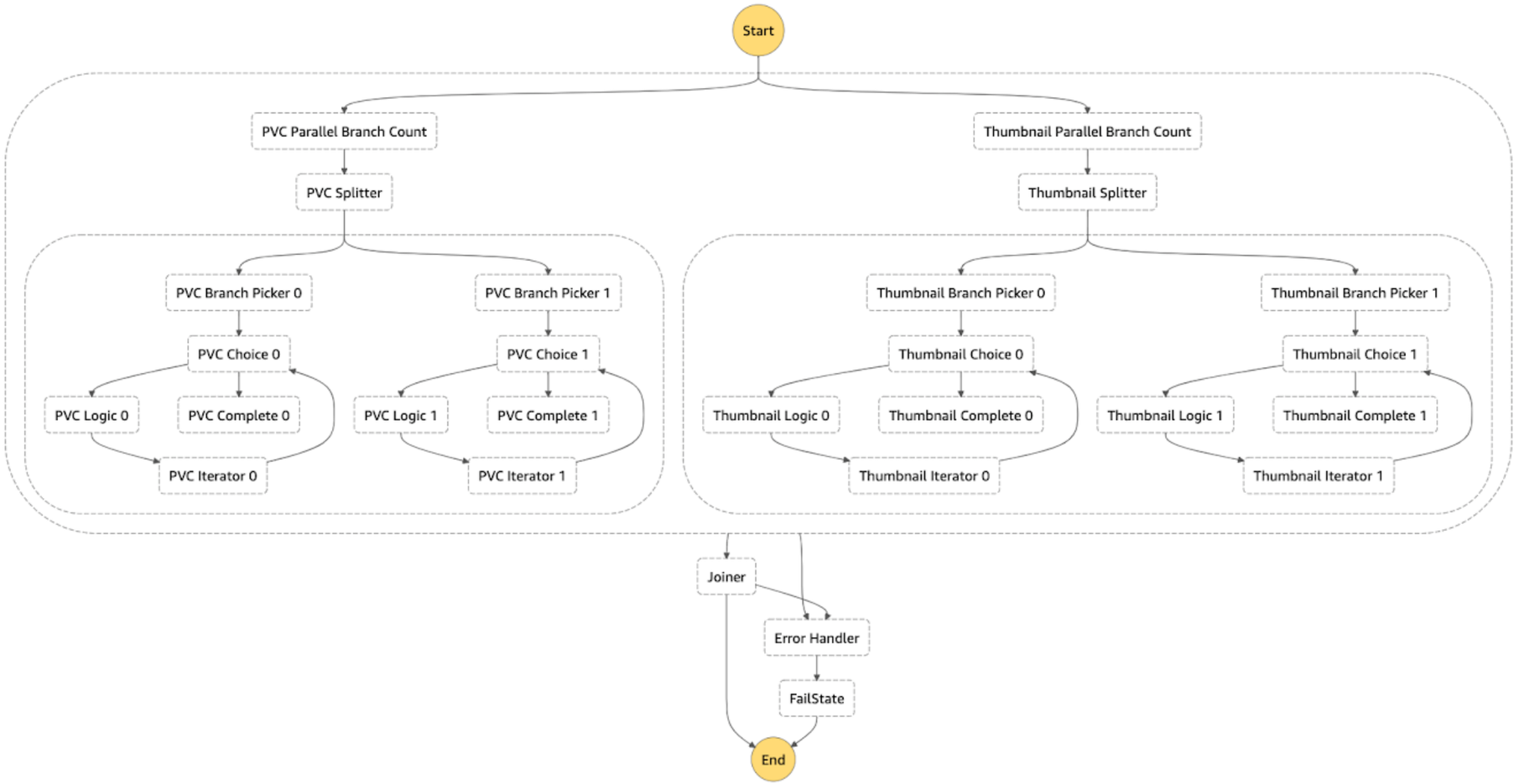
Zillow®

End-to-end for a Single Workflow



Zillow

Combining PVC and Thumbail



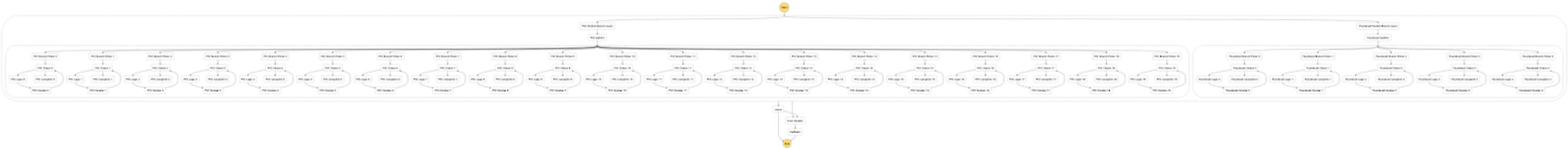
Zillow

After Both Workflows Are Complete

- Joiner Lambda to aggregate results from the two workflows
- Communicates with our Data API to tell it that this task is complete
- Outputs a file in S3 with all the results aggregated
  - We wanted DynamoDB items to be exclusively for within a step function execution
  - TTL on DynamoDB items, but permanent record of task output in S3

Zillow

Scaling Out Branches



Error Handling

- Step Functions allow you to retry a task for certain error types
- Built-in backoff, number of attempts, etc.
- Catch-all or specific errors

Key Takeaway: Always catch AWS service exceptions and retry on them

Key Takeaway 2: Throw custom exceptions in your code for retryable vs non-retryable errors

```
"X": {
   "Type": "Task",
   "Resource": "arn:aws:::example",
   "Next": "Y",
   "Retry": [ {
      "ErrorEquals": [ "ErrorA", "ErrorB" ],
      "IntervalSeconds": 1,
      "BackoffRate": 2.0,
      "MaxAttempts": 2
   }, {
      "ErrorEquals": [ "ErrorC" ],
      "IntervalSeconds": 5
   } ],
   "Catch": [ {
      "ErrorEquals": [ "States.ALL" ],
      "Next": "Z"
   } ]
}
```

Zillow

Monitoring

- Cloudwatch Alarms
  - Thresholds on Lambda failures and other metrics
  - Alarms on Step Function failures
  - Hanging executions
  - Posts a message to SNS on alarm
- Notifications on Alarm
  - Notifies our incident tracking system (OpsGenie)
  - Alerts our monitoring Slack channels

Zillow

Lambda Aliasing

- Give each Lambda an alias when deploying a breaking changes
  - v1, v2, etc.
- Step Function tasks can point to a Lambda's *qualified arn*
  - Will execute the specific alias you tell it to
- When making breaking changes:
  - Ship updated lambdas with their own new alias
  - Step Function will still be executing against the old aliases
  - Switch the Step Function definition all at once to point to the new aliases

Zillow

How Did We Do?

- Compared to our old solution:
  - Responds much better to bursts in traffic
  - Compartmentalized logic for individual tasks, easier to maintain
  - Can choose the parallelization scale for each workflow
- Easy to add additional tasks to perform
  - Add payload generation in splitter
  - Add new iterator loop
  - Add processing in joiner

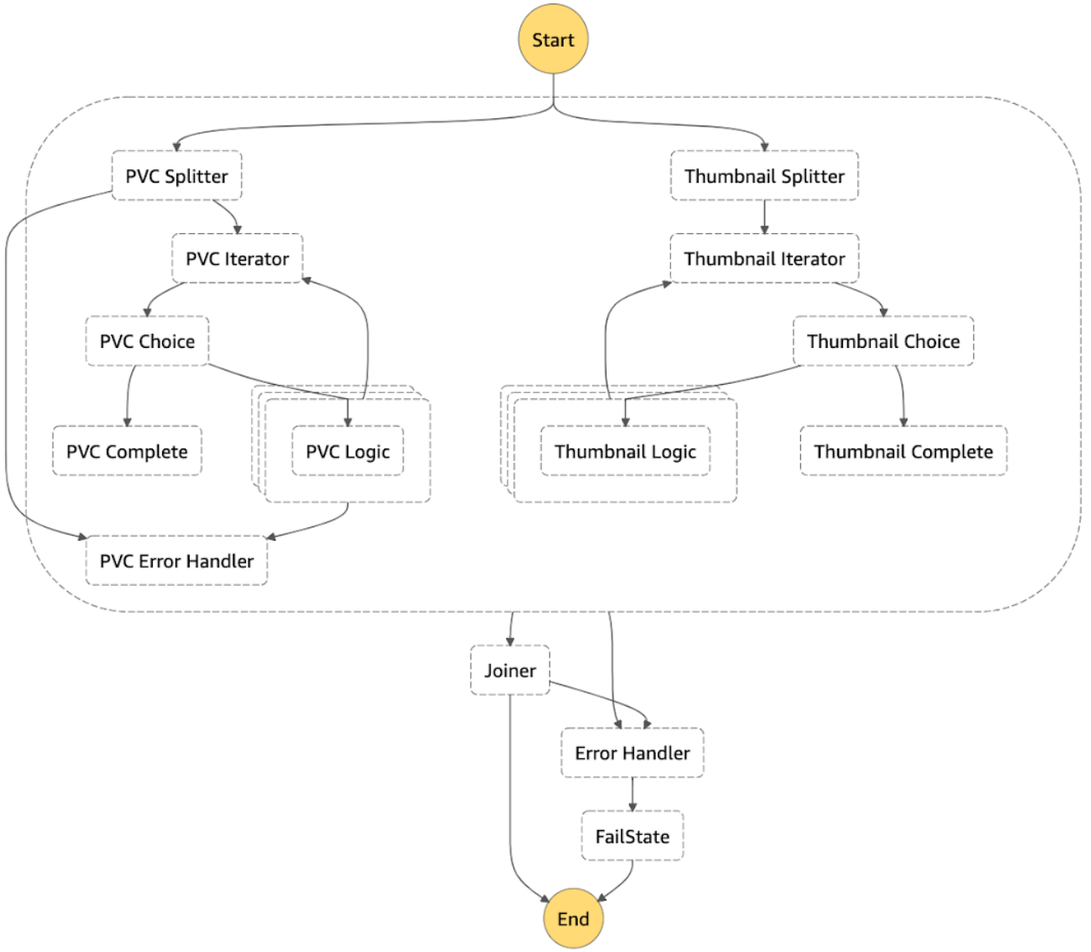
Zillow

Next Steps

- Machine learning with Sagemaker
  - Integrate directly into Step Function to run machine learning models on our images
- Extended our workflow to generate a smart floor map using CV
- Dynamic parallel workflows
  - Provide a list of inputs to the new Map state and allow it to iterate for you
  - Still limited to 32KB, need to investigate further

Zillow

Dynamic Parallelism



Zillow

Demo

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Step Functions

cf831735-fb0b-4fb5-9d08-da48f60e0ad2_1359909

Visual workflow



Thank you!

Nikhil Gupta
nikhilg@zillowgroup.com

Randy Yao
randyy@zillowgroup.com

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws