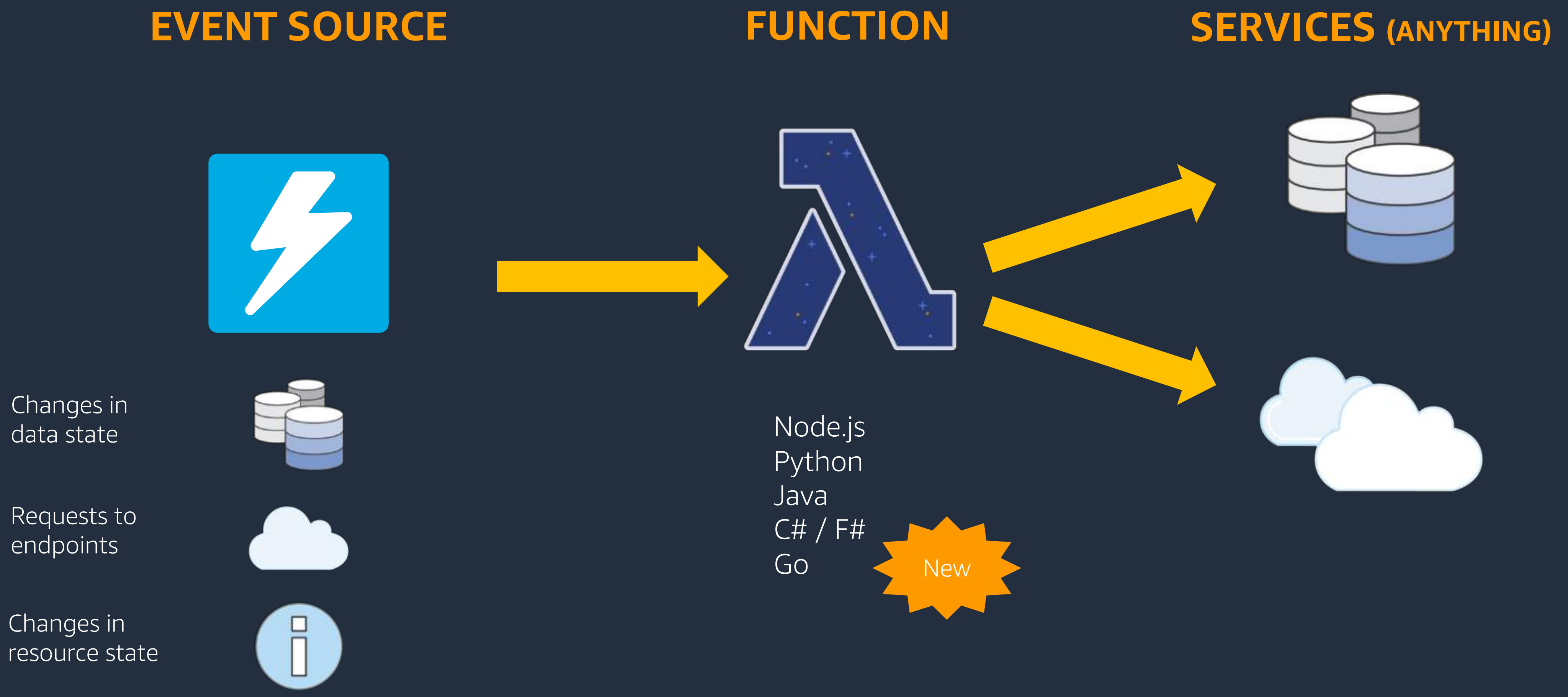
INNOVATE 2018
ONLINE CONFERENCE
DEVELOPER EDITION

Serverless Developer Experience

Danilo Poccia, Evangelist, Serverless

@danilop

Serverless applications



Common serverless use cases

Web applications

- Static websites
- Complex web apps
- Packages for Flask and Express

Backends

- Apps and services
- Mobile
- IoT

Data processing

- Real-time
- MapReduce
- Batch

Chatbots

- Powering chatbot logic

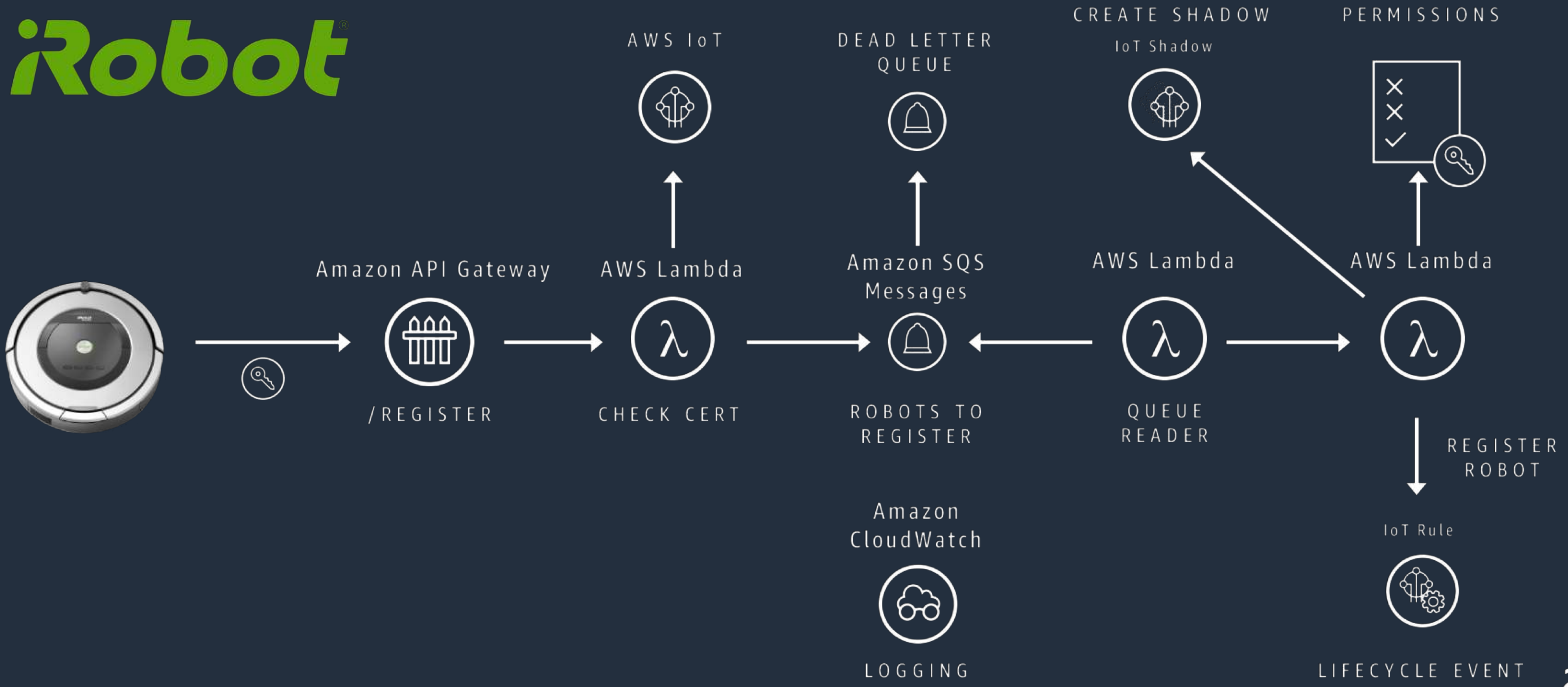
Amazon Alexa

- Powering voice-enabled apps
- Alexa Skills Kit

IT automation

- Policy engines
- Extending AWS services
- Infrastructure management

Case Study



aws

Fannie Mae Serverless Financial Modeling

Case Study

Financial Modeling is a Monte-Carlo simulation process to project future cash flows, which is used for managing the mortgage risk on daily basis:

- Underwriting and valuation
- Risk management
- Financial reporting
- Loss mitigation and loan removal

- ~10 Quadrillion ($10x10^{15}$) of cash flow projections each month in hundreds of economic scenarios.
- One simulation run of ~ 20 million mortgages takes 1.4 hours, >4 times faster than the existing process.

The Federal National Mortgage Association

Smart Resource Allocation

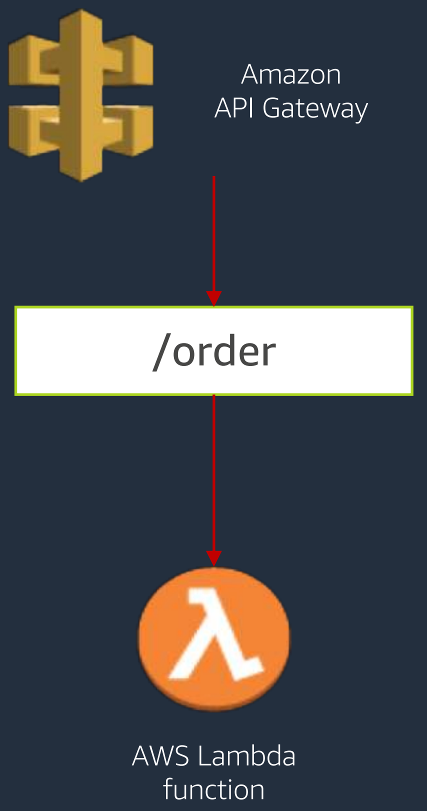
Match resource allocation (up to 3 GB!) to logic

Stats for Lambda function that calculates 1000 times all prime numbers <= 1000000

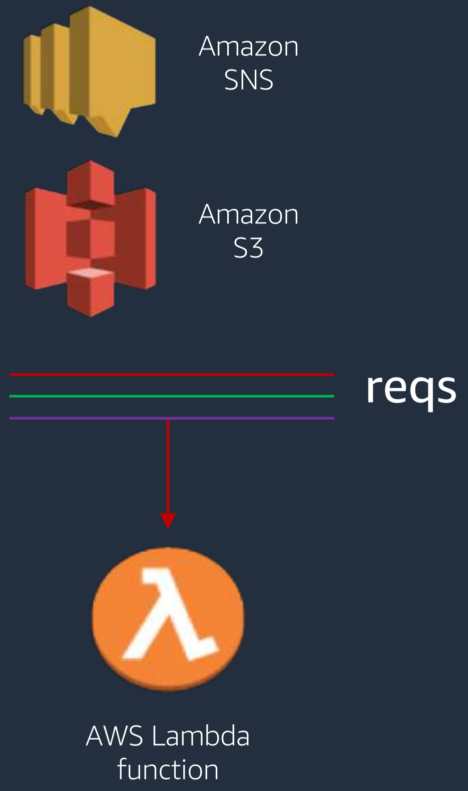
128 MB	11.722965 sec	$0.024628
256 MB	6.678945 sec	$0.028035
512 MB	3.194954 sec	$0.026830
1024 MB	1.465984 sec	$0.024638

Lambda execution model

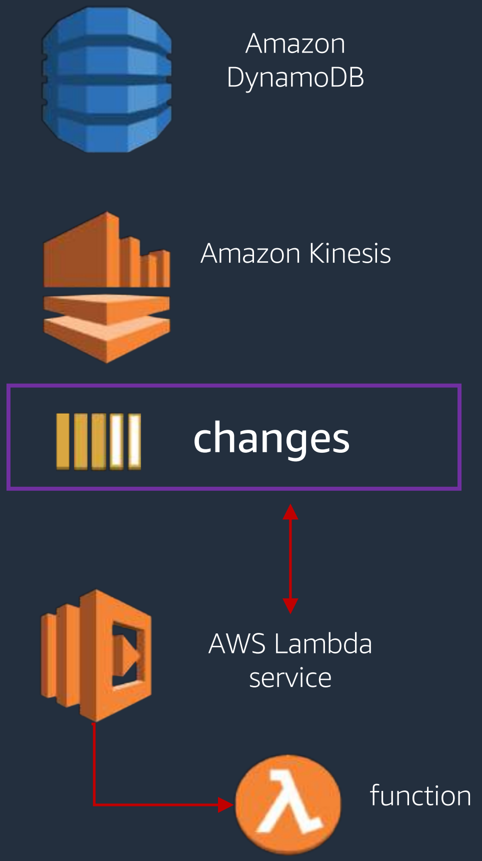
Synchronous (push)



Asynchronous (event)



Stream-based



Lambda permissions model

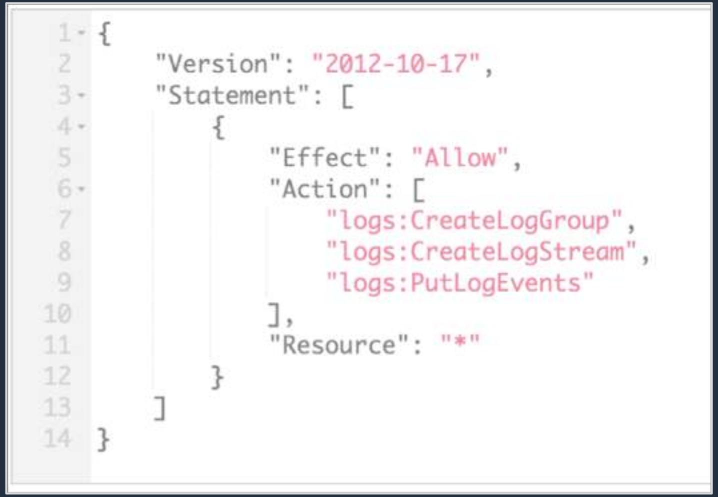
Fine-grained security controls for both execution and invocation

Execution policies:

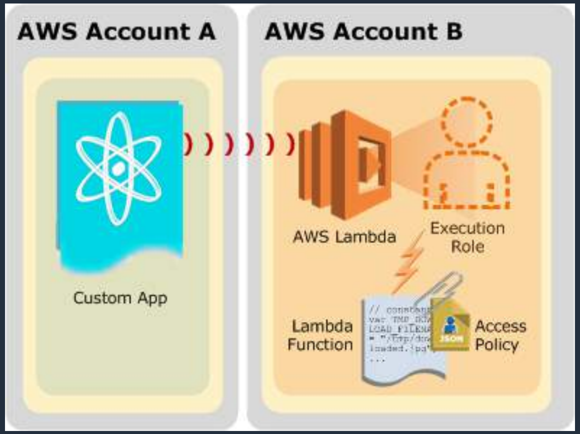
- Define what AWS resources/API calls this function can access via IAM
- Used in streaming invocations
- For example, "Lambda function A can read from DynamoDB table users"

Function policies:

- Used for sync and async invocations
- For example, "Actions on bucket X can invoke Lambda function Z"
- Resource policies allow for cross-account access



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```



Managing Infrastructure as Code

Provision and manage a collection of related AWS resources.

Your application = CloudFormation stack

Input .yaml file and output provisioned AWS resources

Meet SAM!

Serverless Application Model (SAM)

CloudFormation extension optimized for serverless

New serverless resource types: functions, APIs, and tables

Supports anything CloudFormation supports

Open specification (Apache 2.0)

`https://github.com/awslabs/serverless-application-model`

SAM template

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  GetHtmlFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://demo-bucket/todo_list.zip
      Handler: index.js
      Runtime: nodejs6.1
      Policies: AmazonDynamoDBReadOnlyAccess
      Events:
        GetHtml:
          Type: Api
          Properties:
            Path: /{proxy+}
            Method: ANY
```

SAM template

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  GetHtmlFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://demo-bucket/todo_list.zip
      Handler: index.js
      Runtime: nodejs6.1
      Policies: AmazonDynamoDBReadOnlyAccess
      Events:
        GetHtml:
          Type: Api
          Properties:
            Path: /{proxy+}
            Method: ANY
```

```
AWS::Lambda::Function
AWS::IAM::Role
AWS::IAM::Policy

AWS::ApiGateway::RestApi
AWS::ApiGateway::Stage
AWS::ApiGateway::Deployment

AWS::Lambda::Permission
```

CloudFormation template

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  GetHtmlFunctionGetHtmlPermissionProd:
    Type: AWS::Lambda::Permission
    Properties:
      Action: lambda:invokeFunction
      Principal: apigateway.amazonaws.com
      FunctionName:
        Ref: GetHtmlFunction
      SourceArn:
        Fn::Sub: arn:aws:execute-api:${AWS::Region}:${AWS::AccountId}:${ServerlessRestApi}/Prod/ANY/*
  ServerlessRestApiProdStage:
    Type: AWS::ApiGateway::Stage
    Properties:
      DeploymentId:
        Ref: ServerlessRestApiDeployment
      RestApiId:
        Ref: ServerlessRestApi
      StageName: Prod
  ListTable:
    Type: AWS::DynamoDB::Table
    Properties:
      ProvisionedThroughput:
        WriteCapacityUnits: 5
        ReadCapacityUnits: 5
      AttributeDefinitions:
      - AttributeName: id
        AttributeType: S
      KeySchema:
      - KeyType: HASH
        AttributeName: id
  GetHtmlFunction:
    Type: AWS::Lambda::Function
    Properties:
      Handler: index.gethtml
      Code:
        S3Bucket: flourish-demo-bucket
        S3Key: todo_list.zip
      Role:
        Fn::GetAtt:
        - GetHtmlFunctionRole
        - Arn
      Runtime: nodejs4.3
  GetHtmlFunctionRole:
    Type: AWS::IAM::Role
    Properties:
      ManagedPolicyArns:
      - arn:aws:iam::aws:policy/AmazonDynamoDBReadOnlyAccess
      - arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Action:
          - sts:AssumeRole
          Effect: Allow
          Principal:
            Service:
            - lambda.amazonaws.com
  ServerlessRestApiDeployment:
    Type: AWS::ApiGateway::Deployment
    Properties:
      RestApiId:
        Ref: ServerlessRestApi
      Description: 'RestApi deployment id: 127e3fb91142ab1ddc5f5446adb094442581a90d'
      StageName: Stage
  GetHtmlFunctionGetHtmlPermissionTest:
    Type: AWS::Lambda::Permission
    Properties:
      Action: lambda:invokeFunction
      Principal: apigateway.amazonaws.com
      FunctionName:
        Ref: GetHtmlFunction
      SourceArn:
        Fn::Sub: arn:aws:execute-api:${AWS::Region}:${AWS::AccountId}:${ServerlessRestApi}/*/ANY/*
  ServerlessRestApi:
    Type: AWS::ApiGateway::RestApi
    Properties:
      Body:
        info:
          version: '1.0'
          title:
            Ref: AWS::StackName
        paths:
          "/{proxy+}":
            x-amazon-apigateway-any-method:
              x-amazon-apigateway-integration:
                httpMethod: ANY
                type: aws_proxy
                uri:
                  Fn::Sub: arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-
31/functions/${GetHtmlFunction.Arn}/invocations
              responses: {}
        swagger: '2.0'
```

CloudFormation Package/Deploy

```
aws cloudformation package \
    --s3-bucket <BUCKET> \
    --template-file template.yaml \
    --output-template-file packaged.yaml

aws cloudformation deploy \
    --template-file packaged.yaml \
    --stack-name <STACK> \
    --capabilities CAPABILITY_IAM
```

Testing serverless apps - challenges

- Test in an environment that resembles Lambda:
  - OS
  - Libraries
  - Runtime
  - Configured limits (memory, timeout)

- Mimic response and log outputs

Testing serverless apps - challenges

- Test events need to be:
  - Syntactically accurate
  - Different for each trigger

Testing serverless apps - challenges

```
{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "s3": {
        "configurationId": "testConfigRule",
        "object": {
          "eTag": "0123456789abcdef0123456789abcdef",
          "sequencer": "0A1B2C3D4E5F678901",
          "key": "myKey",

"size": 1024
        },
        "bucket": {
          "arn": "arn:aws:s3:::myBucket",
          "name": "myBucket",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          }
        },
        "s3SchemaVersion": "1.0"
      },
      "responseElements": {
        "x-amz-id-2":
"EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwx
yzABCDEFGH",
        "x-amz-request-id": "EXAMPLE123456789"
      },
      "awsRegion": "us-east-1",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "eventSource": "aws:s3” } ] }
```

Introducing the new SAM CLI

```
Usage: sam [OPTIONS] COMMAND [ARGS]...

  AWS Serverless Application Model (SAM) CLI

The AWS Serverless Application Model extends AWS CloudFormation to provide   a
simplified way of defining the Amazon API Gateway APIs, AWS Lambda  functions, and
Amazon DynamoDB tables needed by your serverless application. You can find more in-
depth guide about the SAM specification  here: https://github.com/awslabs/serverless-
application-model.

Options:
  --debug    Turn on debug logging to print debug message generated by SAM CLI.
  --version  Show the version and exit.
  --help     Show this message and exit.

Commands:
  validate  Validate an AWS SAM template.
  init      Initialize a serverless application with a...
  package   Package an AWS SAM application. This is an alias for ‘aws cloudformation
            package'.
  deploy    Deploy an AWS SAM application. This is an alias for ‘aws cloudformation
            deploy'.
  logs      Fetch logs for a function
  local     Run your Serverless application locally for...
```

Introducing the new SAM CLI

```
Usage: sam local [OPTIONS] COMMAND [ARGS]...

  Run your Serverless application locally for quick development & testing

Options:
  --help  Show this message and exit.

Commands:
 generate-event  You can use this command to generate sample...
 invoke          Invokes a local Lambda function once.
 start-api       Sets up a local endpoint you can use to test your API.
                 Supports hot-reloading so you don't need to restart this
                 service when you make changes to your function.
 start-lambda    Starts a local endpoint you can use to invoke your local
                 Lambda functions.
```

Introducing the new SAM CLI

```
https://github.com/awslabs/aws-sam-cli
```

```
pip install --user aws-sam-cli
```

Introducing the new SAM CLI

```
sam init --runtime nodejs --name <NAME>

cd <NAME>/
more README.md
cd hello_world/
more app.js
npm install
cd ..

sam validate

sam local start-api

sam package --template-file template.yaml \
    --s3-bucket <BUCKET> \
    --output-template-file packaged.yaml

sam deploy --template-file packaged.yaml \
    --stack-name <STACK> --capabilities CAPABILITY_IAM
```

INNOVATE 2018

ONLINE CONFERENCE

DEVELOPER EDITION

Demo: AWS SAM CLI

Safe deployments baked into SAM!

New

Lambda aliases now enable traffic shifting

CodeDeploy integration for deployment automation

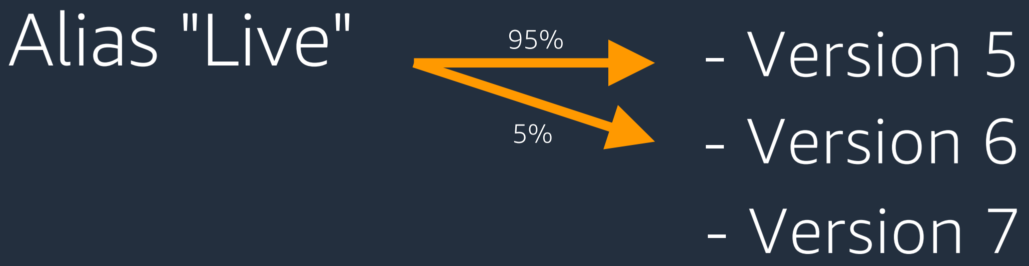
Deployment automation natively supported in SAM

Safe deployments baked into SAM!

New

Version – immutable deployment unit
Alias – pointer to a version

Lambda Function Foo:



Safe deployments baked into SAM!

New

- CodeDeploy integration
  - Preconfigured canary and linear deployments
  - Auto alarm-based rollbacks
  - Pre and post traffic validation hooks
  - Monitor through the CodeDeploy console

- Natively supported in SAM!

Safe deployments baked into SAM!

New

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  GetHtmlFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://demo-bucket/todo_list.zip
      Handler: index.js
      Runtime: nodejs6.1
```

Safe deployments baked into SAM!

New

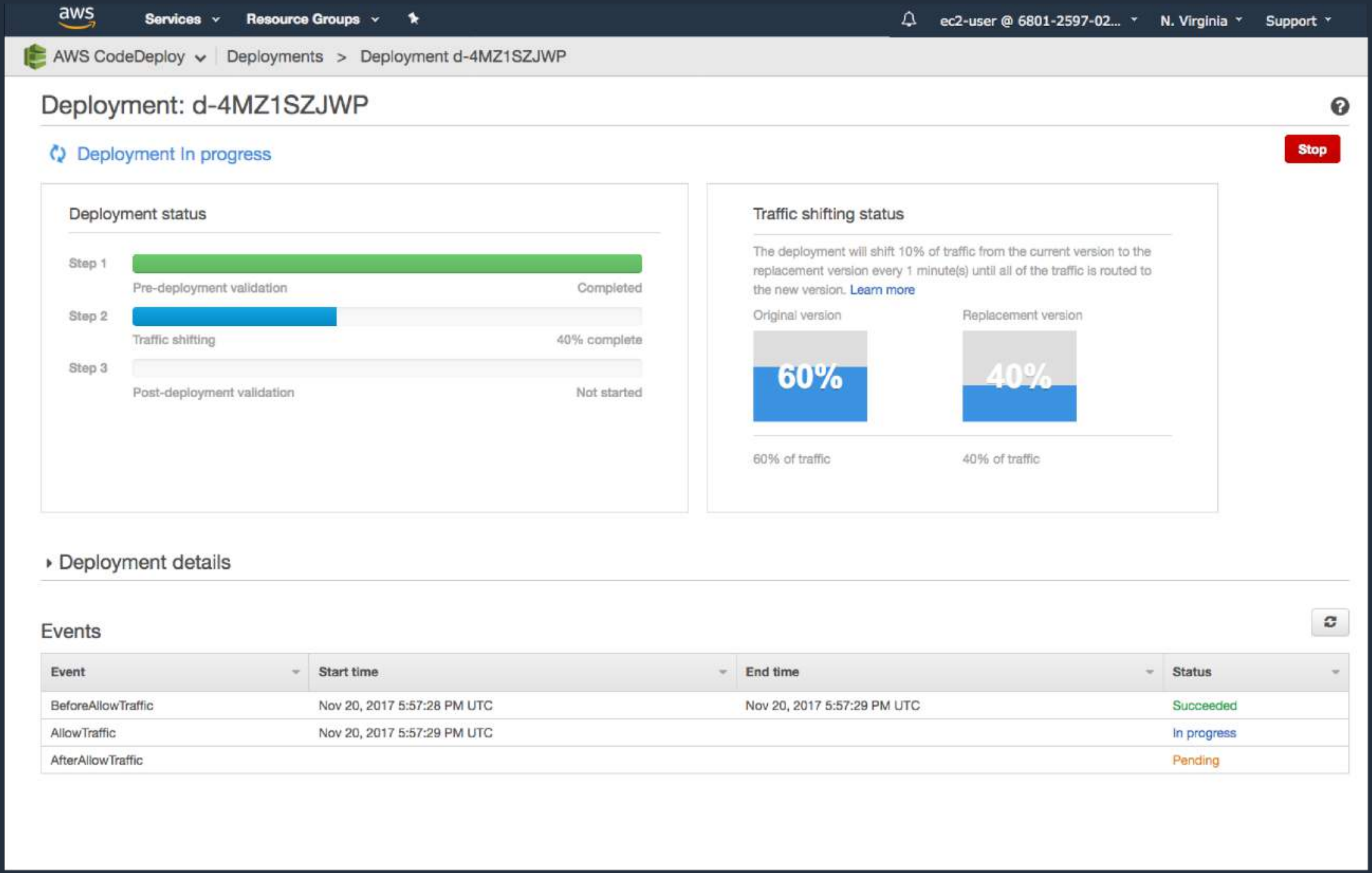
```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Globals:
  Function:
    AutoPublishAlias: Live
    DeploymentPreference:
      Type: Canary10Percent10Minutes
Resources:
  GetHtmlFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://demo-bucket/todo_list.zip
      Handler: index.js
      Runtime: nodejs6.1
      Policies: AmazonDynamoDBReadOnlyAccess
```

Safe deployments baked into SAM!

New

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Globals:
  Function:
    AutoPublishAlias: Live
    DeploymentPreference:
      Type: Canary10Percent10Minutes
Hooks:
        PreTraffic: !Ref CodeDeployHook_PreTest
        PostTraffic: !Ref CodeDeployHook_PostTest
      Alarms:
        - !Ref DurationAlarm
        - !Ref ErrorAlarm
Resources:
  GetHtmlFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://demo-bucket/todo_list.zip
      Handler: index.js
      Runtime: nodejs6.1
      Policies: AmazonDynamoDBReadOnlyAccess
```

Code Deploy console



Deployment: d-4MZ1SZJWP

Deployment In progress

Deployment status

Step 1	Pre-deployment validation	Completed
Step 2	Traffic shifting	40% complete
Step 3	Post-deployment validation	Not started

Traffic shifting status

The deployment will shift 10% of traffic from the current version to the replacement version every 1 minute(s) until all of the traffic is routed to the new version. Learn more

Original version	Replacement version
60% of traffic	40% of traffic

Deployment details

Events

Event	Start time	End time	Status
BeforeAllowTraffic	Nov 20, 2017 5:57:28 PM UTC	Nov 20, 2017 5:57:29 PM UTC	Succeeded
AllowTraffic	Nov 20, 2017 5:57:29 PM UTC		In progress
AfterAllowTraffic			Pending

INNOVATE 2018
ONLINE CONFERENCE

DEVELOPER EDITION

Demo: Safe Deployments

Takeaways

1. Use the **Lambda console** for quick creation and iteration of simple apps
2. Use **AWS SAM** to describe your serverless architecture
3. Plug **SAM CLI** into the IDE of your choice for testing and debugging
4. "Develop in the cloud" with **AWS Cloud9** – optimized for serverless applications
5. Build on SAM for **CI/CD** capabilities, including linear & canary deployments
6. Share your app with the **Serverless Application Repository!**

INNOVATE 2018
ONLINE CONFERENCE
DEVELOPER EDITION

Serverless Developer Experience

Danilo Poccia, Evangelist, Serverless

@danilop