Deploying Serverless Applications

Dror Helper

@dhelper www.helpercode.com

Module Overview

Lambda versioning and aliases

Automating deployment of serverless applications

Running Serverless applications locally

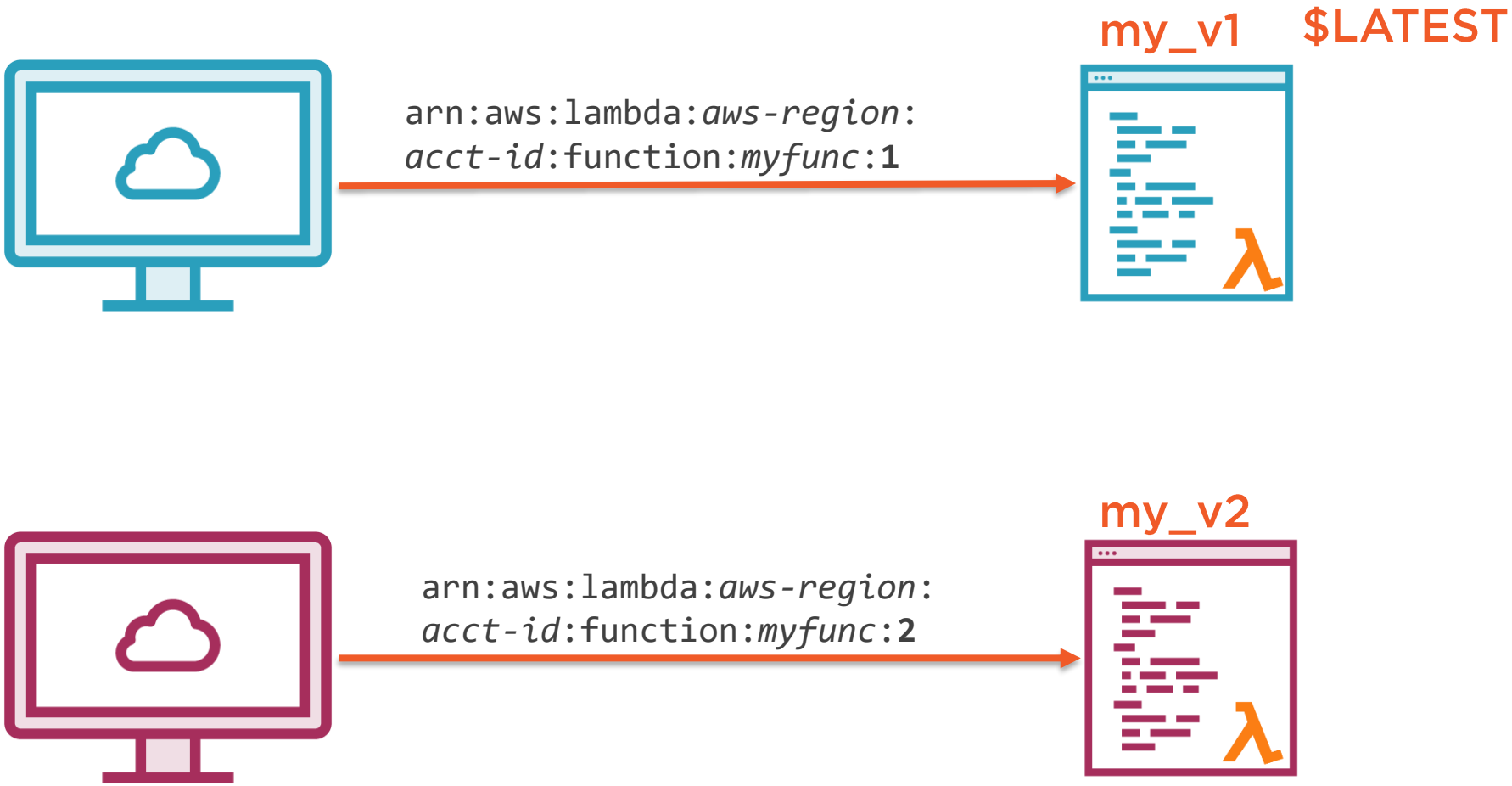
Real World Lambda Challenges

Need to support backward compatibility

Need to separate code by environment

Quickly roll back changes

Lambda Versioning



Lambda Versioning

Can use any string for version name

- Be consistent

It's optional → you decide

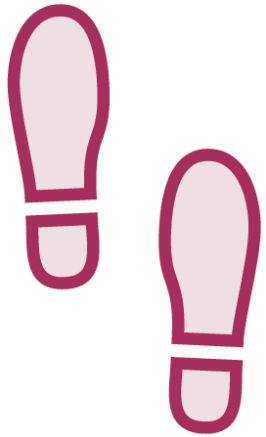
The latest version → $LATEST

- Unqualified version name == $LATEST

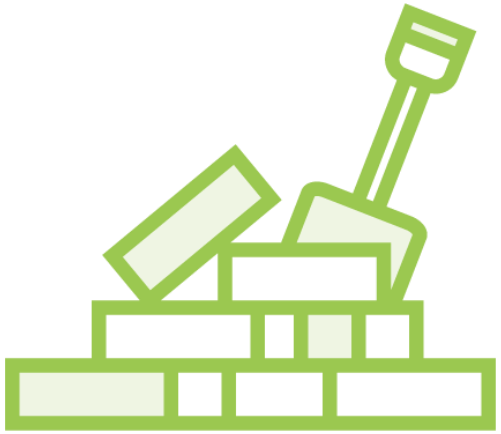
You cannot change code once versioned

Can set permissions per version

Versioning Pain Points



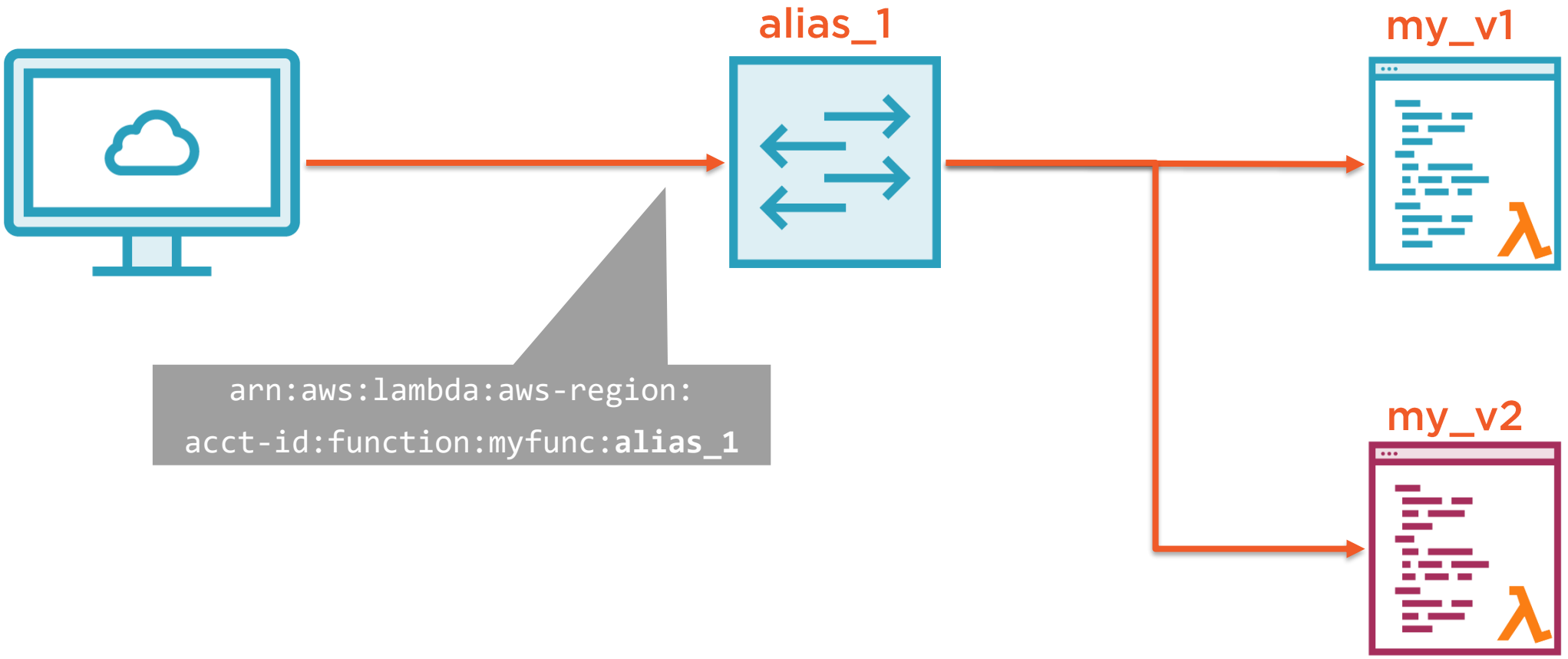
Hard to track which version we should use



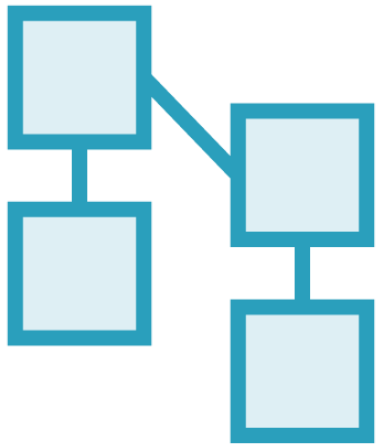
Need to rebuild and deploy calling application

Moving triggers between versions is a pain

Using Aliases



Serverless Application Model



Define serverless applications

- Lambda functions
- Triggers

Natively supported by CloudFormation

Simplified syntax for expressing serverless resources

Hosted on GitHub

- Specifications
- Translation code
- Examples

SAM Specifications

- Resource types
- Event source types
- Properties
- Globals Section

AWS Serverless Model

```
AWSTemplateFormatVersion: '2010-09-09'

Transform: 'AWS::Serverless-2016-10-31'

Description:

Resources:

    CreateItemsFromS3:

        Type: 'AWS::Serverless::Function'

        ...

    Bucket1:

        Type: 'AWS::S3::Bucket'
```

Declaring Lambda Functions

```
Resources:
    CreateItemsFromS3:
    Type: 'AWS::Serverless::Function'
    Properties:
        Handler: lambda_function.lambda_handler
        Runtime: python3.6
        CodeUri: s3://codeBucket/CreateItemsFromS3
        Description: ''
        MemorySize: 128
        Timeout: 180
```

```
Resources:
    CreateItemsFromS3:
    ...
    Events:
        BucketEvent1:
        Type: S3
        Properties:
            Bucket:
                Ref: Bucket1
            Events:
                - 's3:ObjectCreated:*'
            Filter:
                ...
```

Declaring Lambda Triggers

Supported Event Sources

- S3
- SNS
- Kinesis
- DynamoDB
- SQS
- Api
- Schedule
- CloudWatchEvent
- CloudWatchLogs
- IoTRule
- AlexaSkill

```
Globals:
    Function:
        Runtime: Python3.6
        Timeout: 300
Resources:
    Lambda_A: ...
    Lambda_B: ...
```

Globals Section

Define properties common to all your Serverless Functions and APIs

Properties defined in *Globals* inherited by supported resources

- Functions
- Api

Deploying Serverless Applications Using SAM

```
aws cloudformation package \
    --template-file template.yaml \
    --output-template-file output.yaml \
    --s3-bucket my.bucket.name

aws cloudformation deploy \
    --template-file output.yaml \
    --stack-name my-serverless-stack \
    --capabilities CAPABILITY_IAM
```

Introducing SAM CLI

Open source (GitHub)

Additional SAM capabilities

- Initialize as serverless application
- Package an AWS SAM application
- Validate an AWS SAM template
- Deploy an AWS SAM
- SAM Local
  - Run Lambda functions locally
  - Invoke function using “services”
  - Run API gateway locally

Getting SAM CLI

Prerequisites

- Docker (OS dependent)
- Python (2.7)

Install

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

Verify

```
> sam --version
```

```
sam local invoke “TestFunction” -e event.json -t example.yaml
```

```
echo ‘{“data” : “1 2 3” }’ | sam local invoke “TestFunction”
```

Invoking Lambda Functions Locally

Can invoke Lambda by logical ID

- using event file
- Or pass information from stdin

Expects template.yaml (otherwise use –t)

Generating Sample Events

sam local generate-event <service> <args>

Supported sources:

- API Gateway
- DynamoDB
- Kinesis
- S3
- Schedule
- SNS

sam local start-api

Run API Gateway locally

Automatically finds and mounts any functions with “Api” event source

Uses proxy integration by default

Summary

Managing different Lambda deployments

- Versioning
- Aliases

Serverless Aplication Model

- Define serverless applications
- Deployment via CloudFormation
- SAM CLI
  - Running functions locally
  - Generate sample events
  - Run API Gateway locally