Versions and Aliases

In this lesson, you will learn how to create different versions and aliases for your AWS Lambda functions.

WE'LL COVER THE FOLLOWING

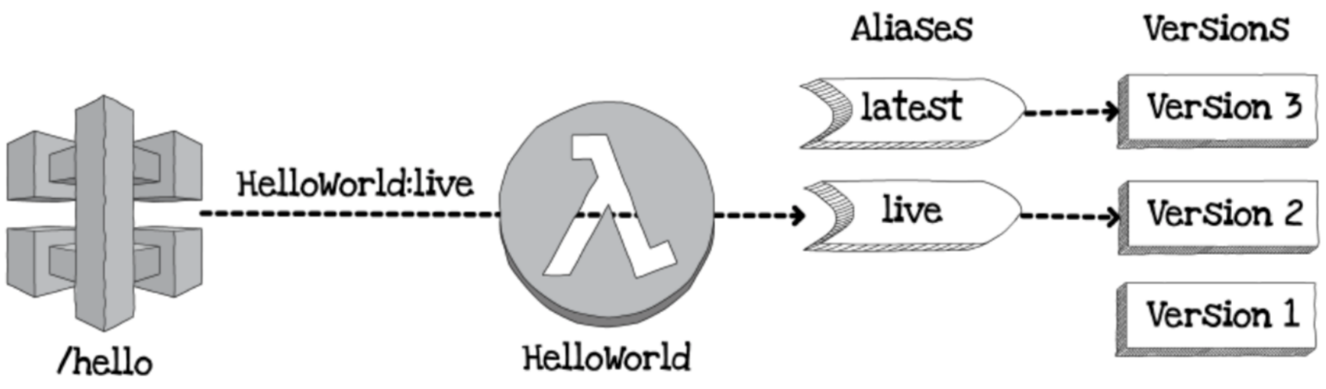
When you deployed the example stack, SAM wired the API gateway to use the `$LATEST` version of your Lambda function. That's OK for a simple case, but it might be problematic for backwards incompatible deployments in the future. Updating a distributed architecture is not instantaneous. You don't want the API to somehow get updated first and then send a new version of an event to an older version of the function. Lambda and API Gateway charge for requests, not for the number of environments, so there is no special cost for keeping an old copy around while the new one is being created. With container-based application hosts, the only solution to this problem would be to create a completely different stack and then switch request routing on the load balancer to the new environment at the end of deployment. With Lambda, it's easy to do that even with a single stack.

Published versions

You can tell Lambda to keep a configured version by publishing it. *Published versions* are read-only copies of function configurations and they are not wiped out after a subsequent update. An event source can request that a particular published function version handles its events. That way, old deployments of the API Gateway can ask for the old Lambda code, and new deployments can ask for the new Lambda function. During an update, any events aimed at the previous version will just keep running on old containers. Once no events have requested an old version for a while, Lambda will remove those instances. When an event targets the new version, Lambda will create a new container using the newly published configuration.

Aliases

To make deployments safe, you need to make sure that events target a particular version, not `$LATEST`. Each published configuration version has a unique numerical ID, assigned by Lambda incrementally with each deployment. In theory, you could manually keep track of these, ensure that event sources request a particular numerical version, and update the template before each deployment to rewire event sources, but this would be error-prone and laborious. Lambda allows you to declare configuration *aliases*, meaningful names pointing to a numerical version. For example, you can create an alias called `live` to represent a published configuration version and set up all event sources to request that alias. After an application update, you do not need to rewire event sources. You just need to point the alias to the new numerical version.



An event source can ask for a specific alias, which points to a numerical version of a function.

SAM automates the whole process of publishing numerical versions, assigning aliases, and wiring event sources to aliases. All you need to do is add the `AutoPublishAlias` property to the function properties, followed by the alias. An alias can be any Latin alphanumeric text between 1 and 128 characters.

Let's make deployments safer by using an alias for the `HelloWorldFunction`. Change `template.yaml` and add `AutoPublishAlias` to the function properties at the same indentation level as `Runtime`. Give the alias a meaningful name. For example, in the listing below, the alias will be `live` (see line 7).

```yaml
HelloWorldFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: hello-world/
    Handler: app.lambdaHandler
    Runtime: nodejs12.x
    AutoPublishAlias: live
```

app/template.yaml

Note that you do not need to change the alias name for each deployment. SAM will automatically publish a new numerical version and then reassign the existing alias to it.

> ### SAM versions or CloudFormation versions?
>
> CloudFormation has a low-level resource for Lambda versions (`AWS::Lambda::Version`), but at the time this was written, it was almost useless. In order to create it, you'd need to specify the SHA256 hash of the latest deployed version, which you generally won't know before deployment. You also can't get it during the deployment of a Lambda function, because the CloudFormation Lambda resource does not expose it in any way. The SAM resource using `AutoPublishAlias` is much simpler and more useful.

Next, you'll create two published versions. When you click on the **Run** button after making the change to `template.yaml`, the widget will run `sam build`, `sam package`, and `sam deploy` with the usual parameters in order to create the first published version of the function configuration.

You can check the output by accessing the URL provided by AWS.

Environment Variables

Key:	Value:
AWS_ACCESS_KEY_ID	Not Specified...
AWS_SECRET_ACCE...	Not Specified...
BUCKET_NAME	Not Specified...

AWS_REGION	Not Specified...

```
{
  "body": "{\"message\": \"hello world\"}",
  "resource": "/{proxy+}",
  "path": "/path/to/resource",
  "httpMethod": "POST",
  "isBase64Encoded": false,
  "queryStringParameters": {
    "foo": "bar"
  },
  "pathParameters": {
    "proxy": "/path/to/resource"
  },
  "stageVariables": {
    "baz": "qux"
  },
  "headers": {
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8",
    "Accept-Encoding": "gzip, deflate, sdch",
    "Accept-Language": "en-US,en;q=0.8",
    "Cache-Control": "max-age=0",
    "CloudFront-Forwarded-Proto": "https",
    "CloudFront-Is-Desktop-Viewer": "true",
    "CloudFront-Is-Mobile-Viewer": "false",
    "CloudFront-Is-SmartTV-Viewer": "false",
    "CloudFront-Is-Tablet-Viewer": "false",
    "CloudFront-Viewer-Country": "US",
    "Host": "1234567890.execute-api.us-east-1.amazonaws.com",
    "Upgrade-Insecure-Requests": "1",
    "User-Agent": "Custom User Agent String",
    "Via": "1.1 08f323deadbeefa7af34d5feb414ce27.cloudfront.net (CloudFront)",
    "X-Amz-Cf-Id": "cDehVQoZnx43VYQb9j2-nvCh-9z396Uhbp027Y2JvkCPNLmGJHqlaA==",
    "X-Forwarded-For": "127.0.0.1, 127.0.0.2",
    "X-Forwarded-Port": "443",
    "X-Forwarded-Proto": "https"
  },
  "requestContext": {
    "accountId": "123456789012",
    "resourceId": "123456",
    "stage": "prod",
    "requestId": "c6af9ac6-7b61-11e6-9a41-93e8deadbeef",
    "requestTime": "09/Apr/2015:12:34:56 +0000",
    "requestTimeEpoch": 1428582896000,
    "identity": {
      "cognitoIdentityPoolId": null,
      "accountId": null,
      "cognitoIdentityId": null,
      "caller": null,
      "accessKey": null,
      "sourceIp": "127.0.0.1",
      "cognitoAuthenticationType": null,
      "cognitoAuthenticationProvider": null,
      "userArn": null,
      "userAgent": "Custom User Agent String",
      "user": null
    },
    "path": "/prod/path/to/resource",
    "resourcePath": "/{proxy+}",
    "httpMethod": "POST",
    "apiId": "1234567890",
    "protocol": "HTTP/1.1"


  }
}
```

Now, to see the difference easily, open `hello-world/app.js` and change the response message (for example, change it to `hello world 2`). Press the **Run** button again to register the changes. After that build, package and deploy the function again.

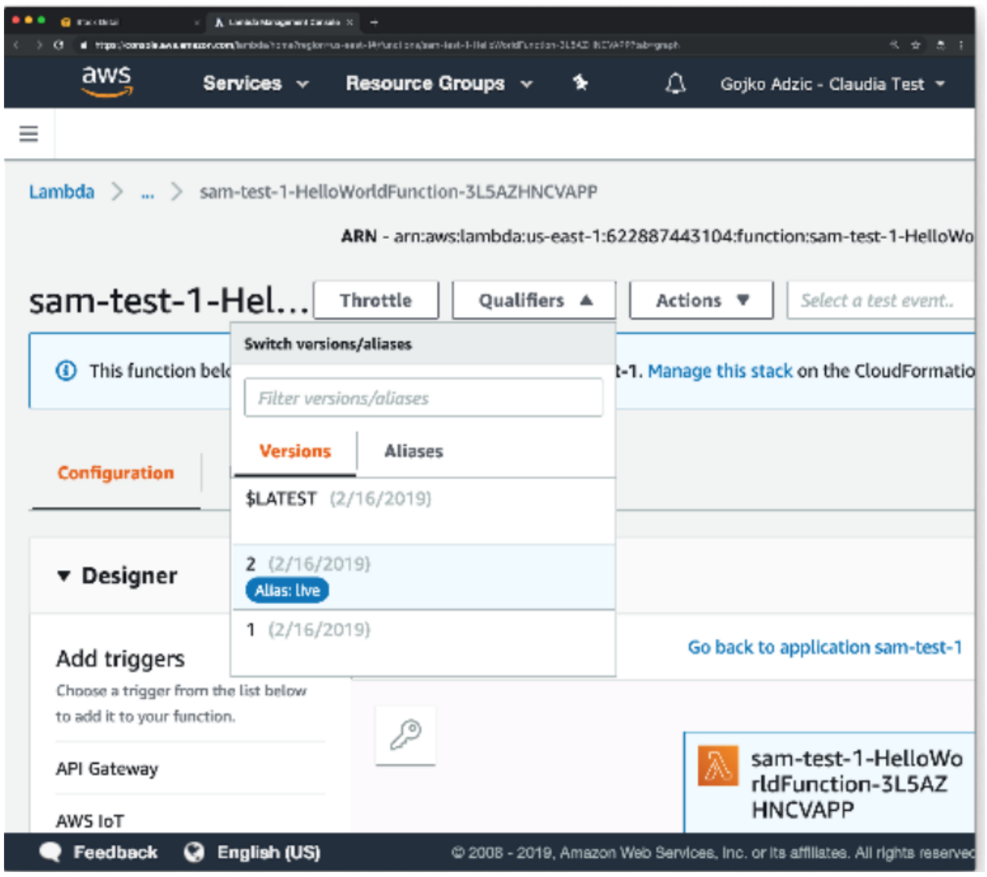
You can do it by running the following set of commands:

```
sam build && \
sam package --s3-bucket $BUCKET_NAME --output-template-file output.yaml && \
sam deploy --template-file output.yaml --stack-name sam-test-1 --capabilities CAPABILITY_IAM
```

You can see the versions and aliases for your Lambda function in the AWS Web Console:

1. Open the AWS Web Console at https://aws.amazon.com and sign in if necessary.
2. Navigate to the Lambda service.
3. Find the `HelloWorld` function and open the function configuration page.
4. Click on the *Qualifiers* drop down and switch to the *Versions* tab within the drop-down list.

You deployed the stack twice, which created two versions, so you should see two published versions of your function in the *Versions* tab, and the alias `live` assigned to the second version.


aws
Services
Resource Groups
Gojko Adzic - Claudia Test
Lambda > ... > sam-test-1-HelloWorldFunction-3L5AZHNCVAPP
ARN - arn:aws:lambda:us-east-1:622887443104:function:sam-test-1-HelloWo
sam-test-1-Hel...
Throttle
Qualifiers
Actions
Select a test event..
Switch versions/aliases
Filter versions/aliases
Versions
Aliases
$LATEST (2/16/2019)
2 (2/16/2019)
Alias: live
1 (2/16/2019)
This function bel
-1. Manage this stack on the CloudFormatio
Configuration
Designer
Add triggers
Choose a trigger from the list below to add it to your function.
API Gateway
AWS IoT
Go back to application sam-test-1
sam-test-1-HelloWorldFunction-3L5AZHNCVAPP
Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved

See the versions and aliases of your Lambda function in the Qualifiers drop-down of the Lambda Web Console.

Checking out invoked versions

You can also get a list of configuration versions from the command line using the following command:

```
aws lambda list-versions-by-function --function-name <NAME>
```

Hint: To get the function name, you must use the following command as specified in the last lesson.

```
aws cloudformation list-stack-resources --stack-name sam-test-1
```

When an event source wants to invoke a function, it can specify a particular version of the configuration it needs either by providing the configuration number or by specifying an alias.

When an `AWS::Serverless::Function` template includes the `AutoPublishAlias` property, SAM will automatically configure all event sources specified inside the resource to request that particular alias. For event sources specified outside the function resource, you can invoke an alias or a specific version by adding a colon and qualifier after the function name. For example, instead of directly targeting `sam-test-1-HelloWorldFunction-QL5VEY42DZ2C`, configure the event source to connect to version 1 by using the target `sam-test-1-HelloWorldFunction-QL5VEY42DZ2C:1`.

By default, an AWS account has 75 GB available for storing Lambda functions, including the code for all published versions. SAM will not automatically clean up old versions for you, so you may need to periodically do some housekeeping if you deploy large packages frequently. On the other hand, any versions that you do not explicitly delete will be instantly available in the future. Unlike application hosts where rolling back to a previous version requires another deployment, switching back and forth between versions with Lambda can be almost instant. You can also use old published versions for as long as you want.

For example, you can invoke the function directly using the following command:

```
aws lambda invoke --function-name <NAME> <OUTPUT FILE NAME>
```

Note that, unlike the `sam local invoke` command used in the previous chapter, `aws lambda invoke` sends an event to the function deployed to AWS. This will execute the Lambda function, print out the execution status, and save the function output in the specified output file. Print out the file contents to see the actual response:

```
$ aws lambda invoke --function-name sam-test-1-HelloWorldFunction-QL5VEY42
DZ2C result.txt
{
 "StatusCode": 200,
 "ExecutedVersion": "$LATEST"
}

$ cat result.txt
{"statusCode":200,"body":"{\"message\":\"hello world 2\"}"}
```

Because you did not provide a specific version, the `ExecutedVersion` (on line 4 of the previous listing) shows `$LATEST`. Now specify the previous version using `--qualifier`:

```
$ aws lambda invoke --function-name sam-test-1-HelloWorldFunction-QL5VEY42
DZ2C --qualifier 1 result.txt
{
    "StatusCode": 200,
    "ExecutedVersion": "1"
}

$ cat result.txt
{"statusCode":200,"body":"{\"message\":\"hello world\"}"}
```

You can now see that the executed version was `1` and that the function response contained the old message.

In the CloudWatch logs, you can see which version was invoked at the start of each request execution (the entry will contain the word `START`). Log streams will also contain the version in square brackets, so you can easily filter all logs for a particular version.

```
$ sam logs -n HelloWorldFunction --stack-name sam-test-1 --filter START
2020-02-16 15:03:02 Found credentials in shared credentials file: ~/.aws/c
redentials
2020/02/16/[$LATEST]b151f67 2020-02-16T14:00:34.908000 START RequestId: 0a
892e26-3e07-4b33-8a64-1ec71bc031a8 Version: $LATEST
2020/02/16/[1]d261bd3a02304 2020-02-16T14:00:50.080000 START RequestId: 32
982750-6d6e-4fc9-a6e0-f00858099291 Version: 1
```

Aliases and deployment pipelines

Although in theory, you can set up different aliases for development, testing and production, this is not easy with CloudFormation and SAM. The more usual pattern is to just set up different stacks. That way each stack has completely separate functions, and those functions each have their own versions aliases.

The SAM `AutoPublishAlias` shortcut is helpful to always reassign the same alias but it does not help much with managing multiple versions in the same stack. Aliases with SAM are mostly useful to make deployments safer.

In the next lesson, you can see how versions and aliases can help you with gradual deployments in the next lesson!