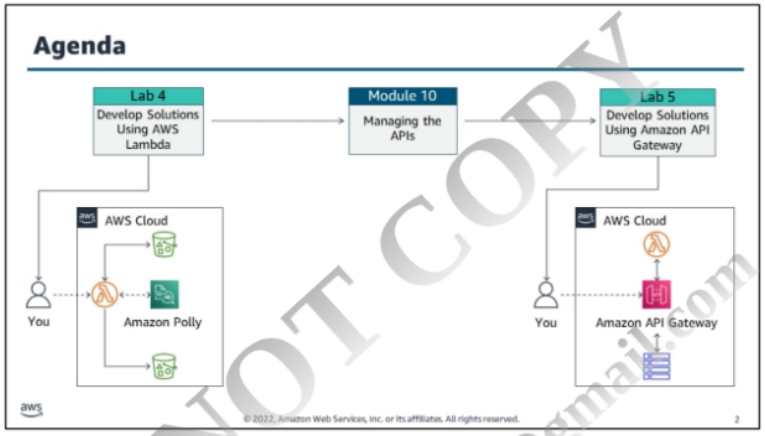
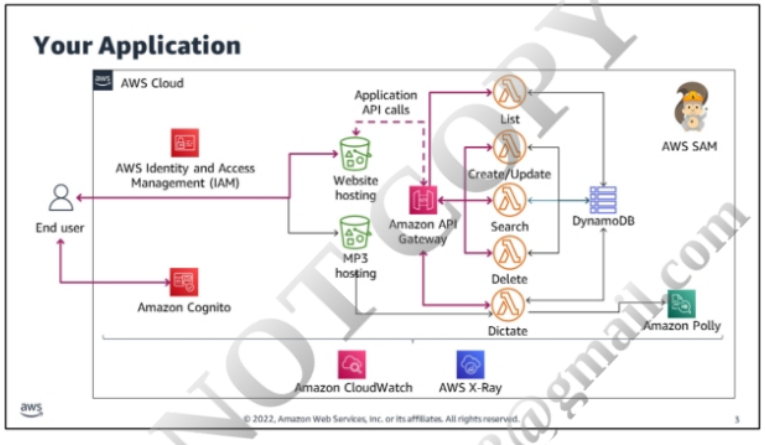
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

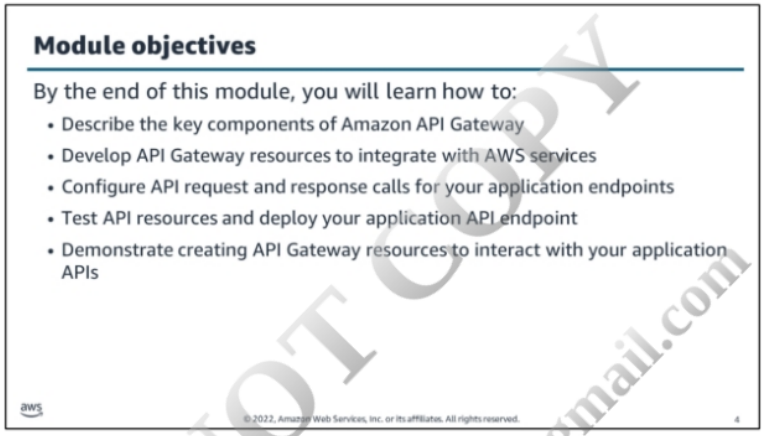




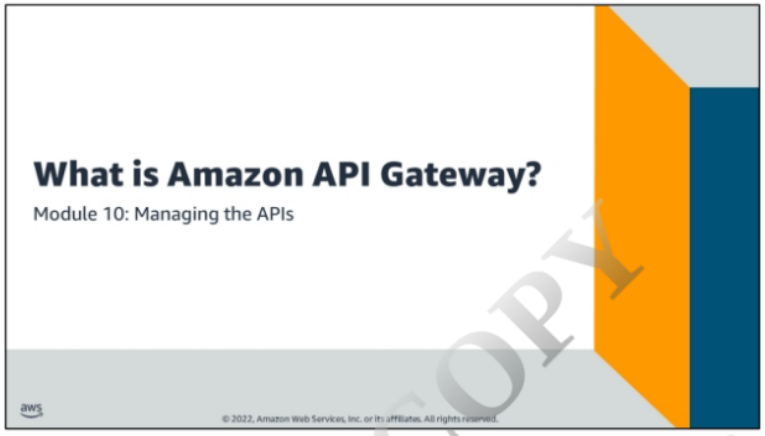
You have built the backend database storage system and configured compute processing. Now you must configure a way for users to access those services over the internet. Amazon API Gateway connects all the application's services together.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 10: Managing the APIs

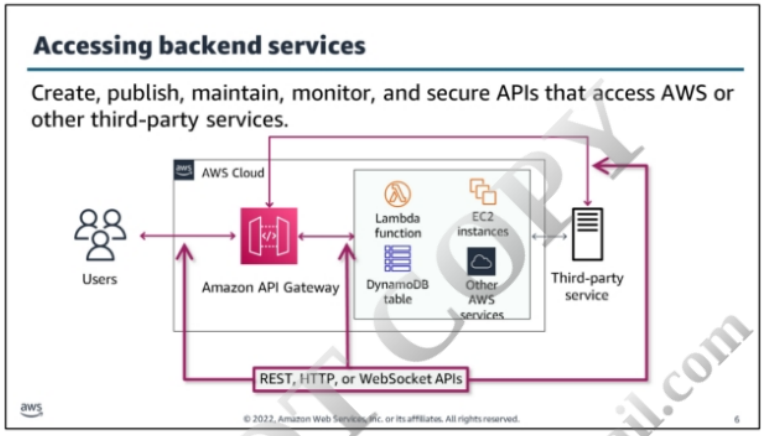


DO NOT COPY
sameersheik68@gmail.com



Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

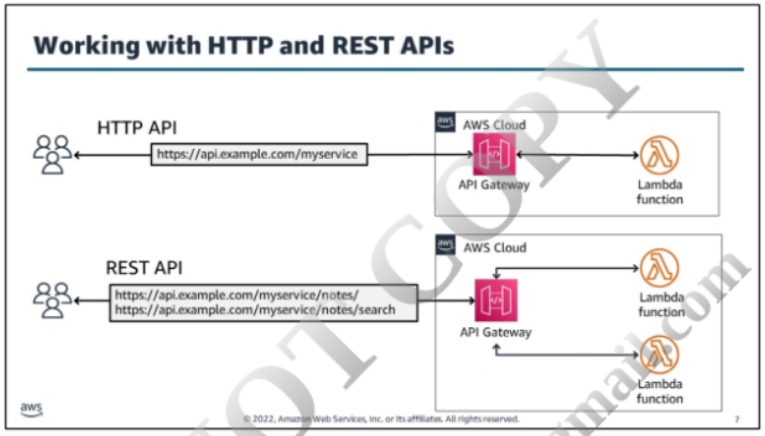
AWS Training and Certification Module 10: Managing the APIs



Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs. Use API Gateway to access AWS services or third-party services.

This diagram illustrates how the APIs you build with API Gateway provide a consistent developer experience for building AWS serverless applications. API Gateway handles all the tasks involved in accepting and processing thousands of concurrent API calls. These tasks include traffic management, authorization and access control, monitoring, and API version management.

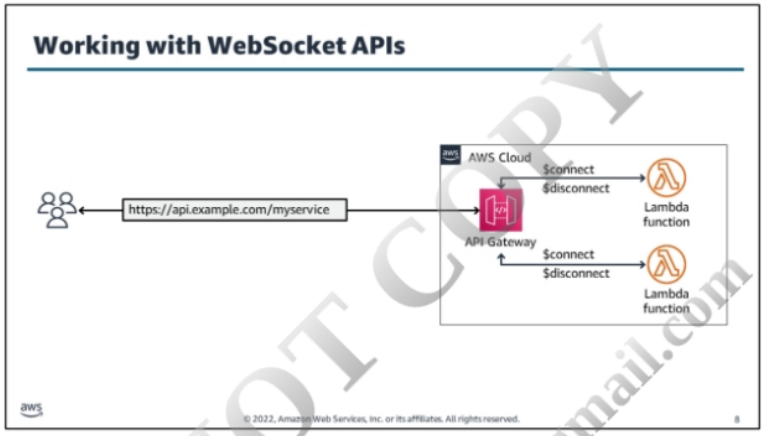
API Gateway acts as a "front door" for applications to access data, business logic, or functionality from your backend services. Examples of this capability are workloads running on Amazon Elastic Compute Cloud (Amazon EC2), code running on AWS Lambda, any web application, or real-time communication applications.



Use HTTP APIs to create RESTful APIs with lower latency and lower cost than REST APIs. For example, you can create an HTTP API that integrates with a Lambda function on the backend. When a client calls your API, API Gateway sends the request to the Lambda function and returns the function's response to the client.

REST APIs are a collection of resources and methods that are integrated with backend HTTP endpoints, Lambda functions, or other AWS services. You can use API Gateway features to help you with all aspects of the API lifecycle, from creation through monitoring your production APIs. Your application will use a REST API.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



With API Gateway, you can create a WebSocket API as a stateful frontend for an AWS service (such as Lambda or DynamoDB) or for an HTTP endpoint. The WebSocket API invokes your backend based on the content of the messages it receives from client applications. In your WebSocket API, incoming JSON messages are directed to backend integrations based on routes that you configure.

Bidirectional APIs

API Gateway WebSocket APIs are bidirectional. A client can send messages to a service, and services can independently send messages to clients. This bidirectional behavior enables richer client/service interactions because services can push data to clients without requiring clients to make an explicit request. WebSocket APIs are often used in real-time applications such as chat applications, collaboration platforms, multiplayer games, and financial trading platforms.

Routes

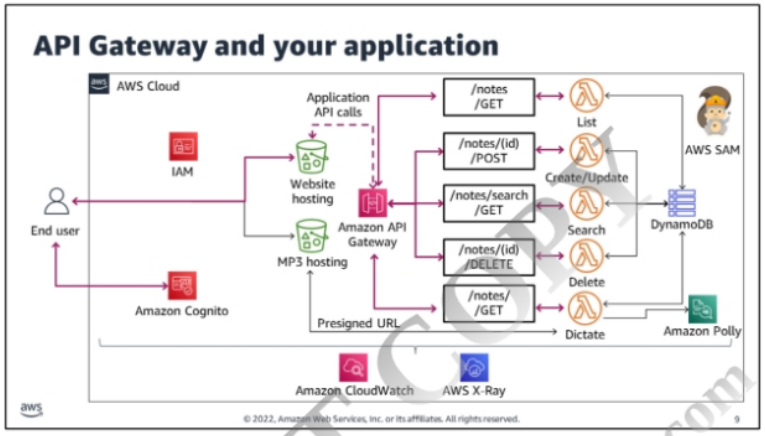
You can use three predefined routes: $connect, $disconnect, and $default. In addition, you can create custom routes. A *route* includes a *route key*, which is the value that is expected once a *route selection expression* is evaluated. The `routeSelectionExpression` is an attribute defined at the API level. It specifies a JSON property that is expected to be present in the message payload. A *route selection expression* is evaluated when the service is selecting the route to follow for an incoming message. The service uses the route whose `routeKey` exactly matches the evaluated value.

The API Gateway calls a route under the following conditions:

- **$connect route** – When a persistent connection between the client and a WebSocket API is being initiated.
- **$disconnect route** – When the client or the server disconnects from the API.
- **Custom route** – After the route selection expression is evaluated against the message if a matching route is found. The match determines which integration is invoked.
- **$default route** – If the route selection expression cannot be evaluated against the message or if no matching route is found.

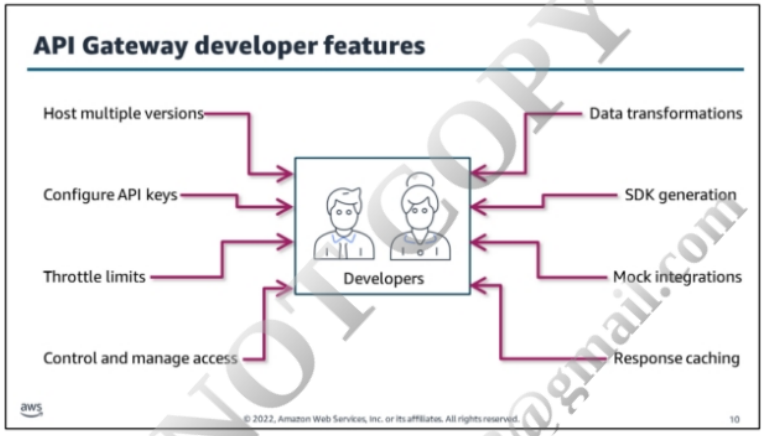
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 10: Managing the APIs



A REST API in API Gateway is a collection of HTTP resources and methods that are integrated with backend HTTP endpoints, Lambda functions, or other AWS services. You can deploy this collection in one or more stages.

Typically, API resources are organized into one or more API methods that have unique HTTP verbs that API Gateway supports. With these methods, GET, POST, and DELETE, you will validate the requests and transform the responses returned from the Lambda functions.



As a developer, API Gateway provides many features to enhance your applications functionality.

Stages
After creating an API, developers can set up a stage. A stage is a named reference to a deployment, which is a snapshot of the API. By using stages, you can do the following:
- Configure stage settings to enable caching
- Customize request throttling
- Configure logging
- Define stage variables
- Attach a canary release for testing

For more information, see "Setting up a stage for a REST API" in the *Amazon API Gateway Developer Guide* (https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-stages.html).

API keys
After you create, test, and deploy your APIs, you can use API Gateway usage plans to make them available as product offerings for your customers. You can configure usage plans and API keys to allow customers to access selected APIs. Customers can access these APIs at agreed-upon request rates and quotas that meet their business requirements and budget constraints.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 10: Managing the APIs

For more information, see "Creating and using usage plans with API keys" in the *Amazon API Gateway Developer Guide* (https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-api-usage-plans.html).

Throttling– To prevent your API from being overwhelmed by too many requests, Amazon API Gateway provides tools to adjust throttle limits to your API. API Gateway provides two basic types of throttling-related settings:
- *Server-side throttling limits* are applied across all clients. These limit settings exist to prevent your API—and your account—from being overwhelmed by too many requests.
- *Per-client throttling limits* are applied to clients that use API keys associated with your usage policy as client identifier.

For more information, see "Throttle API requests for better throughput" in the *Amazon API Gateway Developer Guide* (https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-request-throttling.html).

Access control – API Gateway supports multiple mechanisms for controlling and managing access to your API. Solutions include standard IAM roles and policies, resource policies, cross-origin resource sharing (CORS), and Lambda authorizers.

For more information, see "Controlling and managing access to a REST API in API Gateway" in the *Amazon API Gateway Developer Guide* (https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-control-access-to-api.html).

Data transformations – In API Gateway, an API's method request can take a payload in a different format from the corresponding integration request payload, as required in the backend. Similarly, the backend may return an integration response payload different from the method response payload, as expected by the frontend.

For more information, see "Setting up data transformations for REST APIs" in the *Amazon API Gateway Developer Guide* (https://docs.aws.amazon.com/apigateway/latest/developerguide/rest-api-data-transformations.html).

SDK generation – API Gateway supports generating an SDK for your language-specific application. Supported languages are Java, JavaScript, Java for Android, Objective-C or Swift for iOS, and Ruby.

For more information, see "Generating an SDK for a REST API in API Gateway" in the *Amazon API Gateway Developer Guide* (https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-generate-sdk.html).

Mock integrations – API developers can generate API responses from API Gateway directly, without the need for an integration backend. You decide how API Gateway responds to a mock integration request.

For more information, see "Set up mock integrations in API Gateway" in the *Amazon API Gateway Developer Guide* (https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-mock-

integration.html).

Response caching – Enabling API caching in API Gateway can reduce the number of calls made to your endpoint. Caching can also improve the latency of requests to your API.

For more information, see "Enabling API caching to enhance responsiveness" in the *Amazon API Gateway Developer Guide* (https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-caching.html).

Latency
Each service integrated with an application introduces latency. Using API Gateway, developers with enhanced observability tools can reduce their application's latency.

For more information, see "Troubleshooting Amazon API Gateway with enhanced observability variables" in the AWS Compute blog (https://aws.amazon.com/blogs/compute/troubleshooting-amazon-api-gateway-with-enhanced-observability-variables/).

Security
Security is a shared responsibility between AWS and you. AWS Identity and Access Management (IAM) includes features that are available to use with API Gateway. For example, with identity-based and resource-based policies, you can specify which API operations are allowed or denied on specified resources. You can also specify conditions under which the operations are allowed or denied.

Using Amazon API Gateway, you can create private REST APIs. These REST APIs can be accessed only from your virtual private cloud (VPC) in Amazon VPC by using an interface VPC endpoint. You can use VPC endpoint policies for private APIs in API Gateway. VPC endpoint policies are IAM resource policies that you can attach to an interface VPC endpoint to control access to the endpoint.

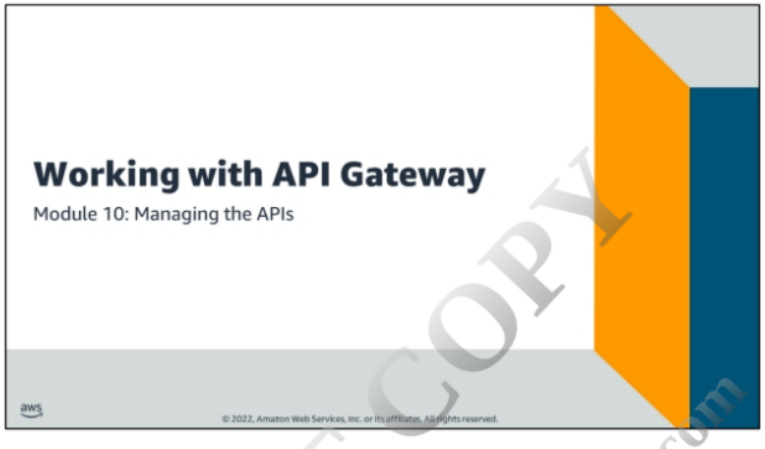
For more information, see "Security in Amazon API Gateway" in the *Amazon API Gateway Developer Guide* (https://docs.aws.amazon.com/apigateway/latest/developerguide/security.html).

Controls
Use API Gateway to control you application at the API level. Tools include throttling to help limit applications from being overwhelmed by too many requests. With AWS WAF integration as the web application firewall, you can protect the applications from attacks.

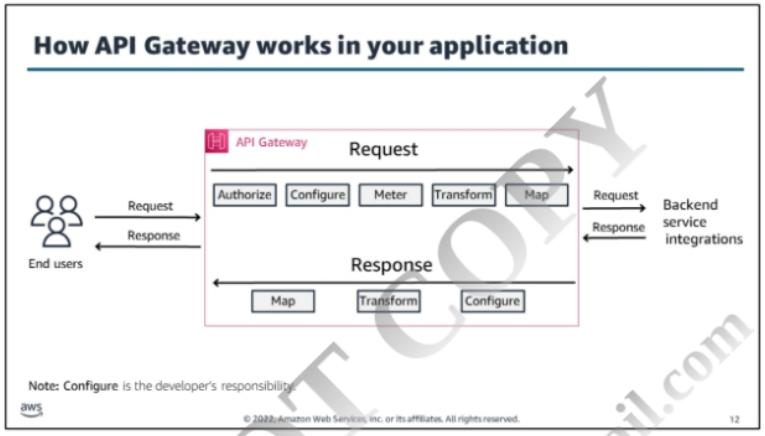
For more information, see "Throttle API requests for better throughput" in the *Amazon API Gateway Developer Guide* (https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-request-throttling.html).

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 10: Managing the APIs



© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.



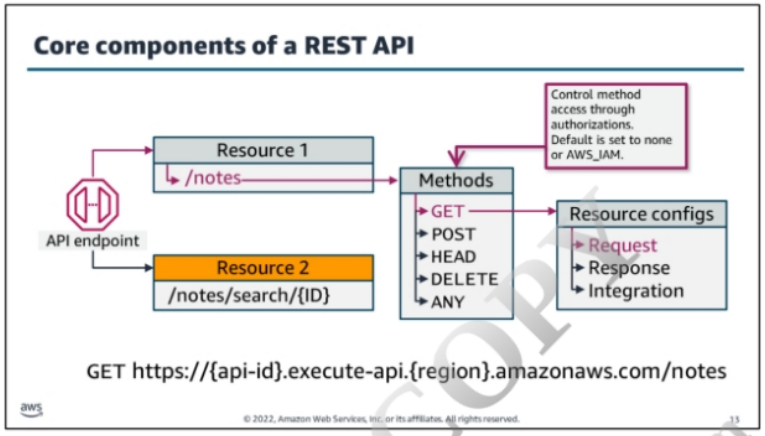
When creating an API method, you must integrate it with an endpoint in the backend. A backend endpoint is also referred to as an *integration endpoint*. An integration endpoint can be a Lambda function, an HTTP webpage, or an AWS service action.

Setting up integration requests
Setting up an integration request involves the following:
- Checking the message for any authorizations
- Configuring how to pass client-submitted method requests to the backend
- Configuring how to transform the request data, if necessary, to the integration request data
- Specifying which Lambda function to call
- Specifying which HTTP server to forward the incoming request to or specifying the AWS service action to invoke

Setting up integration responses (applicable to nonproxy integrations only)
Setting up an integration response involves the following:
- Configuring how to pass the backend-returned result to a method response of a given status code
- Configuring how to transform specified integration response parameters to preconfigured method response parameters
- Configuring how to map the integration response body to the method response body according to the specified body-mapping templates

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 10: Managing the APIs



An API Gateway REST API is made up of resources and methods. A *resource* is a logical entity that an application can access through a resource path. A *method* corresponds to a REST API request that is submitted by the user of your API and the response returned to the user.

API endpoint – A hostname for an API in API Gateway that is deployed to a specific Region. The hostname is of the form {*api-id*}.execute-api.{*region*}.amazonaws.com.

The following types of API endpoints are supported:
- **Regional** – Regional APIs are deployed in the current AWS Region. When API requests predominantly originate from an EC2 instance or services within the same Region where the API is deployed, a regional API endpoint typically lowers the latency of connections. Regional is recommended for such scenarios. This is the default selection for **Endpoint Type**.

- **Edge-optimized** – Edge-optimized APIs are deployed to the Amazon CloudFront network. An edge-optimized API endpoint optimizes access to an API by being deployed to an Amazon CloudFront distribution that is geographically closer to the client.
- **Private** – Private APIs are accessible only from Amazon virtual private clouds (VPCs) using an interface VPC endpoint.

Resources – A collection of HTTP resources and methods that are integrated with backend HTTP endpoints, Lambda functions, or other AWS services. You can deploy this collection in one or more stages. In this example, the API resources are listed as /notes, /notes/search, and /notes/list.

Methods – Represents a client-facing interface by which the client calls the API to access backend resources. Methods include GET, POST, HEAD, DELETE, or any other method of choice. A *method* resource is integrated with an integration resource and consists of a request and one or more responses. You can control method access through authorizations. Default is set to none or set to AWS_IAM. The following are valid values for Authorization:

- **NONE** – No client authentication is performed, and the method is open for access to any user.
- **AWS_IAM** – This selection causes API Gateway to use IAM permissions to control client access to the method. Only users in an IAM role with permission to access the method are allowed to call it. Specifically, the client must sign the request with an AccessKey and corresponding SecretKey as provisioned by IAM to invoke the method. To do this, the client must support the Signature Version 4 (SigV4) protocols.

To require an API key to invoke the method, set **API Key Required** to **true**. To use a usage plan to throttle client traffic, enable API keys.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 10: Managing the APIs

Types of integrations

	Initial Setup	Request and Response Configurations	Passthrough	Services
Proxy	Flexible, versatile, and streamlined integration setup	You do not set the integration request or the integration response.	No options to modify passthrough behaviors	AWS_PROXY HTTP_PROXY
Non-proxy	You are responsible for data mappings	You must configure both the integration request and integration response	You can choose passthrough behaviors	AWS HTTP
Mock	Useful for API testing	Returns a response without sending request to the backend	MOCK	MOCK

 14

Integrations connect a route to backend resources. HTTP APIs support the following:

- Lambda proxy
- AWS service
- Private resources in a VPC
- Mock and HTTP proxy integrations

AWS – Integrate the API method request with an AWS service action, including the Lambda function-invoking action. With the Lambda function-invoking action, this is referred to as the Lambda custom integration. With any other AWS service action, this is known as *AWS integration*.

AWS_PROXY – Integrate the API method request with the Lambda function-invoking action with the client request passed through as-is. This integration is also referred to as the *Lambda proxy integration*.

HTTP – Integrate the API method request with an HTTP endpoint, including a private HTTP endpoint within a VPC. This integration is also referred to as the *HTTP custom integration*.

HTTP_PROXY – Integrate the API method request with an HTTP endpoint, including a private HTTP endpoint within a VPC, with the client request passed through as-is. This is also referred to as the *HTTP proxy integration*.

MOCK – Integrate the API method request with API Gateway as a "loop-back" endpoint without invoking any backend.

With nonproxy integrations, you can choose to pass the client-supplied request payload through the integration request to the backend, without transformation, under the following conditions:

- When a method request carries a payload and the Content-Type header does not match any specified mapping template
- When no mapping template is defined

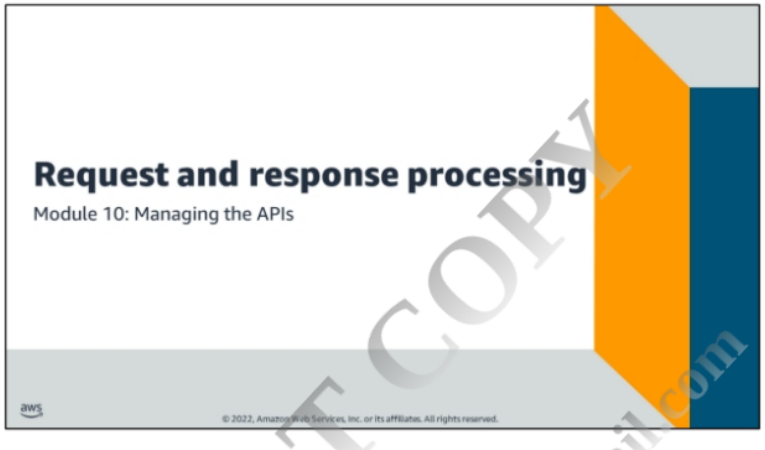
The process is known as *integration passthrough*.

For proxy integrations, API Gateway passes the entire request through to your backend. You do not have the option to modify the passthrough behaviors.

For more information, see "Integration passthrough behavior" in the *Amazon API Gateway Developer Guide* (https://docs.aws.amazon.com/apigateway/latest/developerguide/integration-passthrough-behaviors.html).

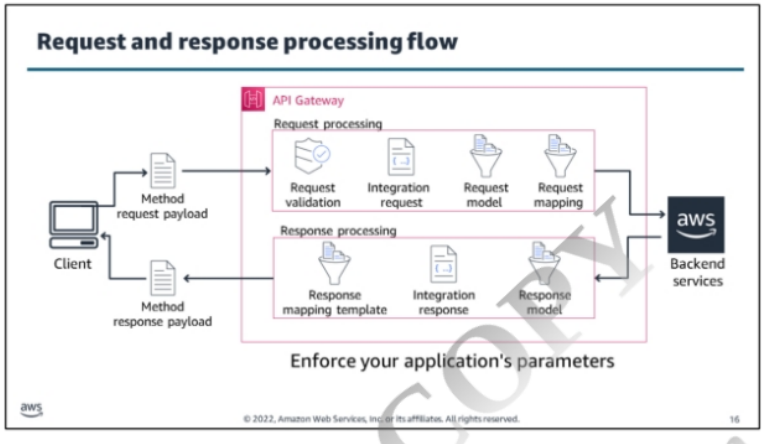
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 10: Managing the APIs



DO NOT COPY sameersheik68@gmail.com

© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.

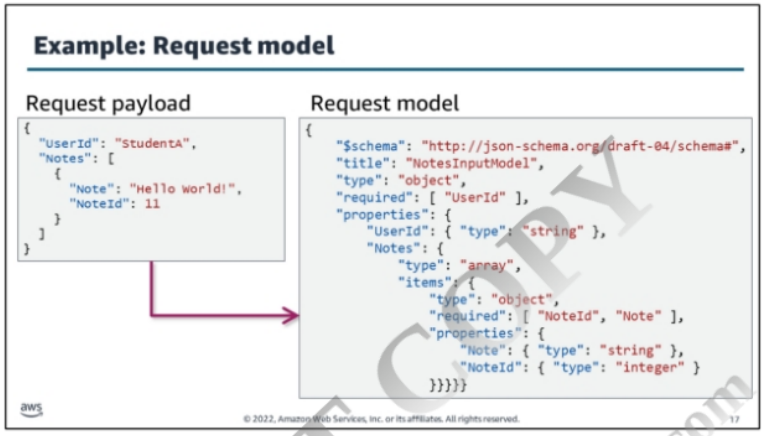


Request and response mapping helps your application transform data to enforce your applications parameters. In API Gateway, an API method request can take a payload in a different format from the corresponding integration request payload, as required in the backend. Similarly, the backend may return an integration response payload different from the method response payload, as expected by the front end.

With API Gateway, you can use mapping templates to map the payload in the following ways:

- From a method request to the corresponding integration request
- From an integration response to the corresponding method response

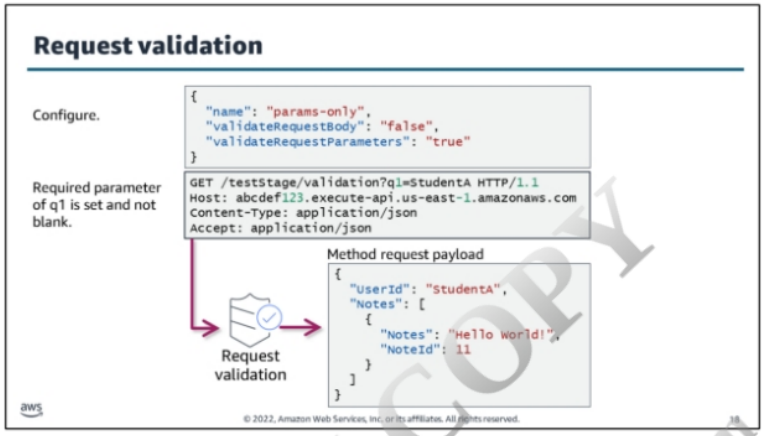
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



Model defines the data structure of a payload.

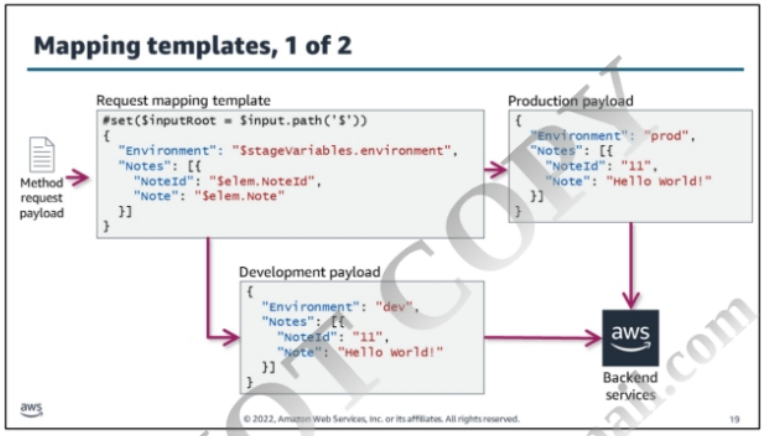
A *mapping template* is a script expressed in Velocity Template Language (VTL) and applied to the payload using JSONPath expressions. A JSONPath expression is for a JSON field of the body of a request or a response. Use mapping templates to do the following:

- Map parameters one-to-one.
- Map a family of integration response status codes (matched by a regular expression) to a single response status code.



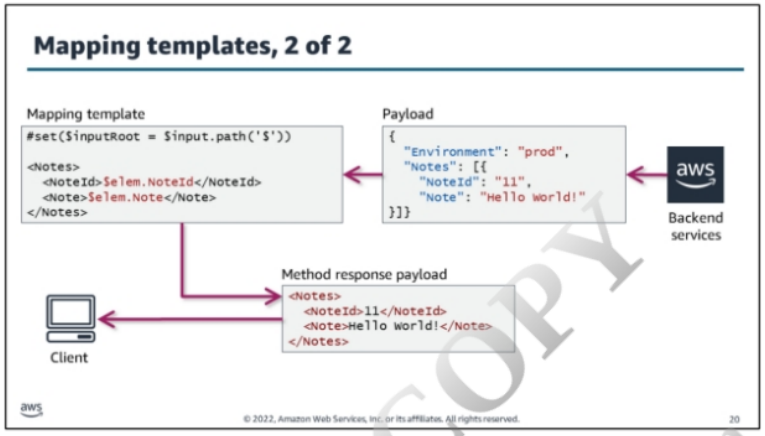
You can configure API Gateway to perform validation of an API request before proceeding.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



Model defines the data structure of a payload.

The mapping template is a script expressed in Velocity Template Language (VTL) and applied to the payload using JSONPath expressions



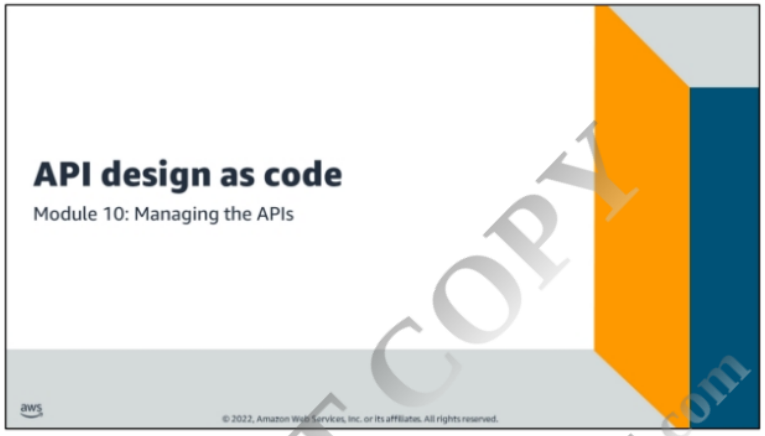
Model defines the data structure of a payload.

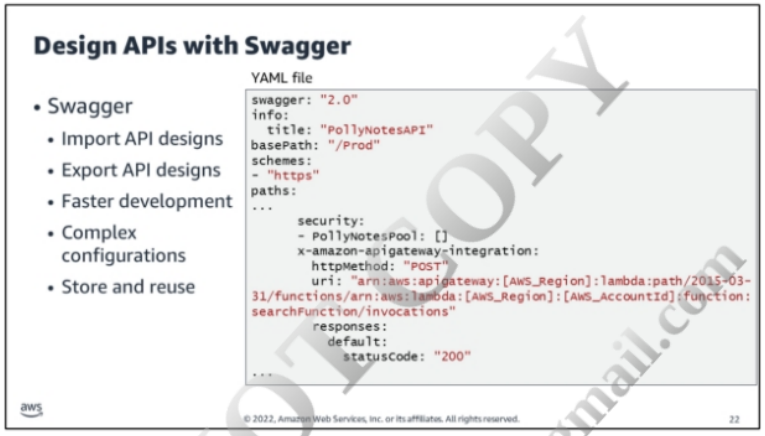
Mapping template is a script expressed in Velocity Template Language (VTL) and applied to the payload using JSONPath expressions.

Remember that the payload includes information about the environment (prod). However, the mapping template strips that information out because it is not needed in the response payload. In this example the mapping template converts the JSON payload to an XML file.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 10: Managing the APIs

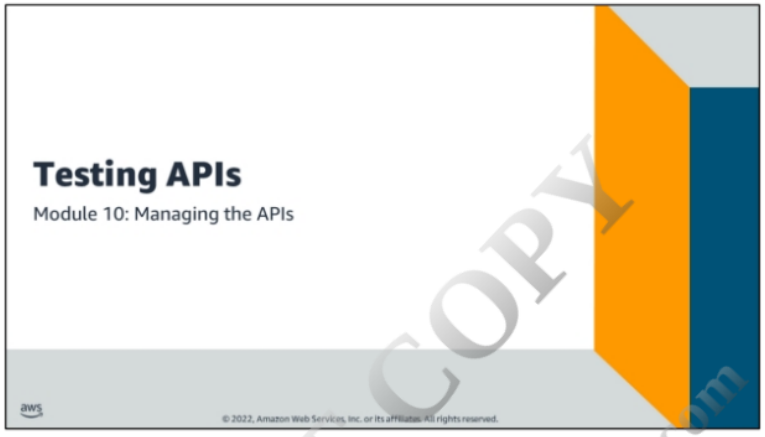


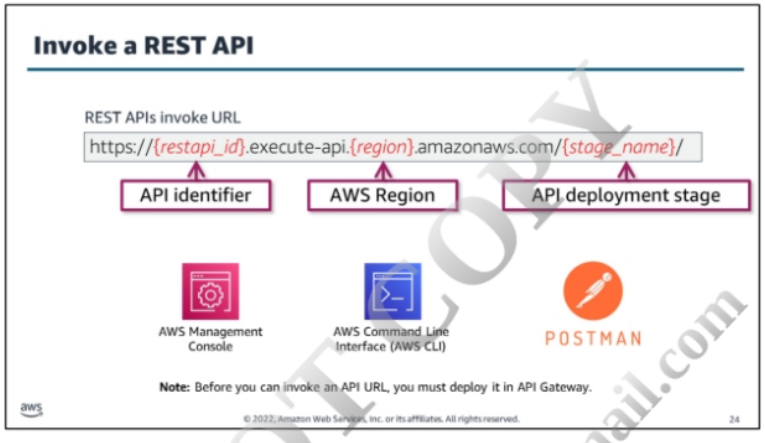


Swagger features
- API definition as code
- Portable API definition
- JSON/YAML
- Import/Export your API
- API Gateway extensions
- Can be used independently or part of the AWS CloudFormation template
- Rich third-party resources, if tools are used

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 10: Managing the APIs





You can test a REST API in various ways. You can use the REST API's invoke URL, the API Gateway console, or third-party tools such as Postman.

If an API permits anonymous access, you can use any web browser to invoke any GET method calls. You can do this by copying an appropriate invocation URL to the browser's address bar. To call a deployed API, clients submit requests to the URL for the API Gateway component service for API operation, known as `execute-api`.

The URL for REST APIs follows this format:

`https://{restapi_id}.execute-api.{region}.amazonaws.com/{stage_name}/`

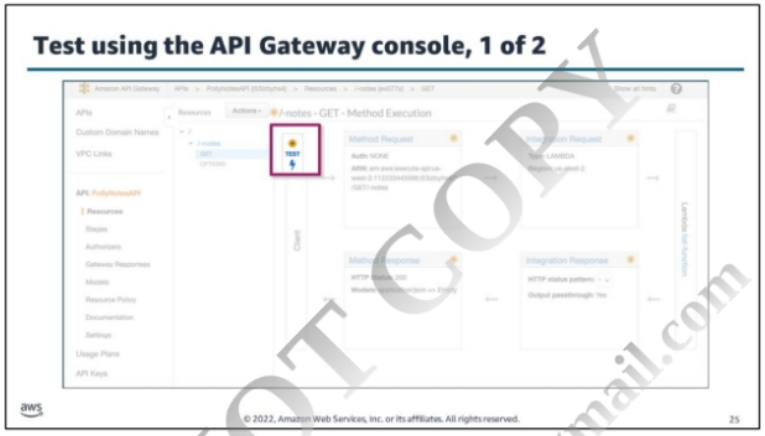
where:
`{restapi_id}` is the API identifier
`{region}` is the Region
`{stage_name}` is the stage name of the API deployment

You can find a REST API's root URL in the Stage Editor for the API in the API Gateway console.

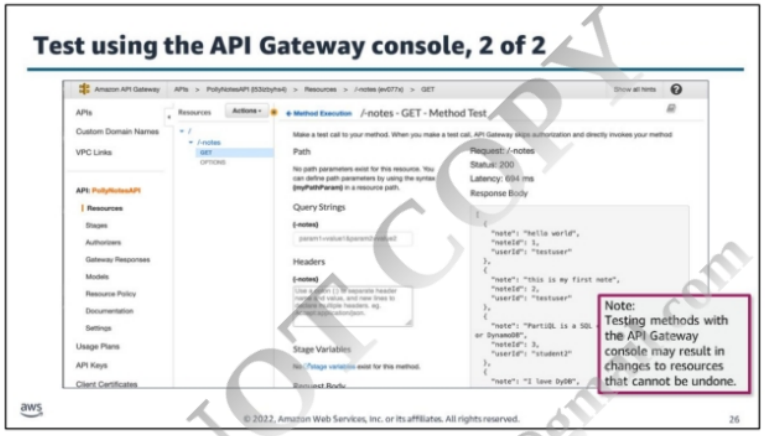
You can use the API Gateway console to call an API by using the API Gateway TestInvoke feature. TestInvoke bypasses the invoke URL and allows API testing before the API is deployed.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 10: Managing the APIs



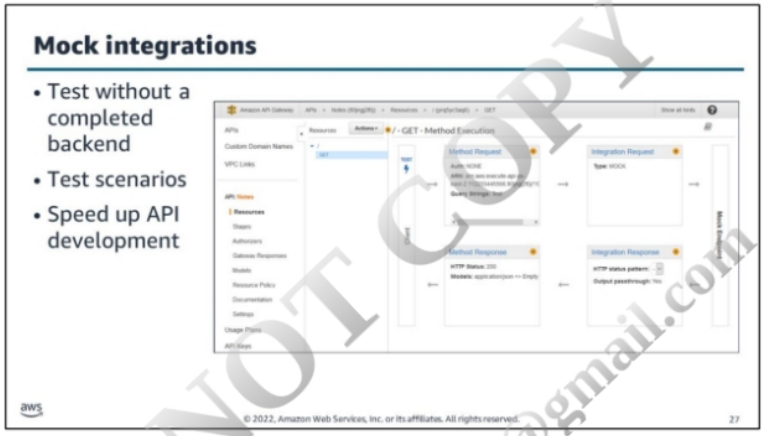
To invoke a REST API from the API Gateway console, in the **Method Execution** pane, in the **Client** box, choose **TEST**.



When the test is completed, the API Gateway console responds with a status of the request.

Testing methods with the API Gateway console may result in changes to resources that cannot be undone. Testing a method with the API Gateway console is the same as calling the method outside of the API Gateway console. For example, if you use the API Gateway console to call a method that deletes an API's resources, if the method call is successful, the API's resources are deleted.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



API developers can generate API responses from API Gateway directly, without the need for an integration backend. As an API developer, you can use this feature to unblock dependent teams that need to work with an API before the project development is complete. You can also test multiple scenarios with the API before deploying it into production.

With mock integrations, API developers are no longer held back by waiting for other teams to complete their integrations.

For more information, see "Set up mock integrations in API Gateway" in the Amazon *API Gateway Developer Guide* (https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-mock-integration.html).

Using the CLI to invoke a REST API

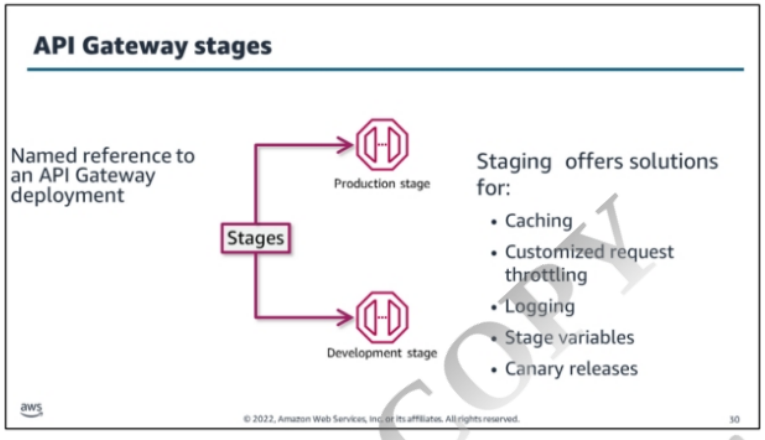
```
Terminal                                                    x
>> aws apigateway test-invoke-method --rest-api-id 81jpgj2f0j --resource-id Prq5yc5aq6 --http-
method GET --path-with-query-string '/'
{
    "status": 200,
    "body": "",
    "headers": {
        "Content-Type": "application/json"
    },
    "multiValueHeaders": {
        "Content-Type": [
            "application/json"
        ]
    },
    "log": "Execution log for request c6b6f73a-1\nWed Aug 18 19:48:34 UTC 2021 : Starting execution for
request: c6b6f73a-1\nWed Aug 18 19:48:34 UTC 2021 : HTTP Method: GET, Resource Path: /\nWed Aug 18 19:48:34
UTC 2021 : Method request path: {}\nWed Aug 18 19:48:34 UTC
2021 : Method request body before transformations: \nWed Aug 18 19:48:34 UTC 2021 : Method response body
after transformations: \nWed Aug 18 19:48:34 UTC 2021 : Method response headers: {Content-
Type=application/json}\nWed Aug 18 19:48:34 UTC 2021 : Successfully completed execution\nWed Aug 18 19:48:34
UTC 2021 : Method completed with status: 200\n",
    "latency": 12
}
```

 28

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Deploying APIs

Module 10: Managing the APIs



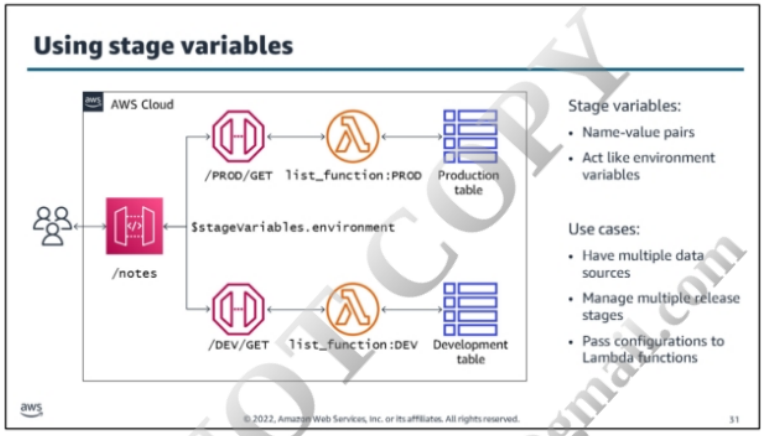
Now that you have configured and tested your API, it is time to deploy it into production and then iterate new versions. To deploy an API, create an API deployment and associate it with a stage. A *stage* is a logical reference to a lifecycle state of your API (for example, dev, prod, beta, v2).

When the deployment process is finished, the **Stage Editor** pane is displayed. The editor shows the URL to invoke the API's `GET /` method request in the **Invoke URL** field.

Using the **Stage Editor**, you can also manage and optimize a particular deployment. For example, you can set up stage settings to enable caching, customize request throttling, configure logging, or define stage variables.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 10: Managing the APIs

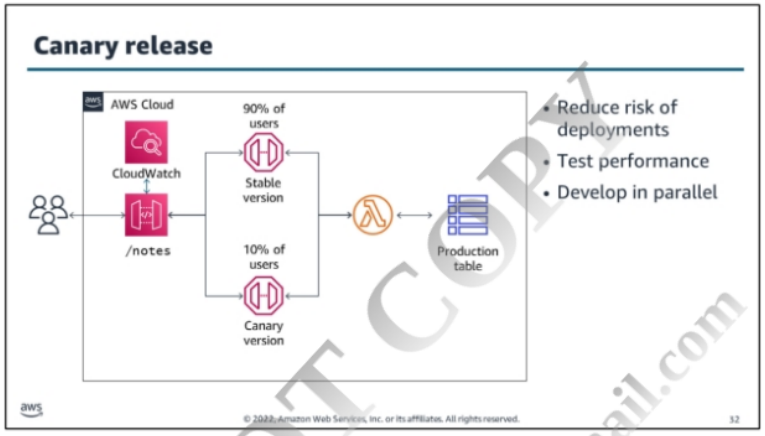


Stage variables, commonly called key-value pairs, are name-value pairs that you can define as configuration attributes. These attributes are associated with a deployment stage of a REST API. They act like environment variables and can be used in your API setup and mapping templates.

For example, you can define a stage variable in a stage configuration. You can then set its value as the URL string of an HTTP integration for a method in your REST API. Later, you can reference the URL string by using the associated stage variable name from the API setup. By doing this, you can use the same API setup with a different endpoint at each stage by resetting the stage variable value to the corresponding URLs. In the example diagram, staging variables can be used to direct traffic to a production and development environment, each with their own DynamoDB tables.

In the diagram, `"/notes"` invokes `"list_function:{$stageVariables.environment}"`. Then API Gateway directs the request, depending on the given environment variable, toward unique data sources.

For more information, see "Setting up stage variables for a REST API deployment" in the *Amazon API Gateway Developer Guide* (https://docs.aws.amazon.com/apigateway/latest/developerguide/stage-variables.html).



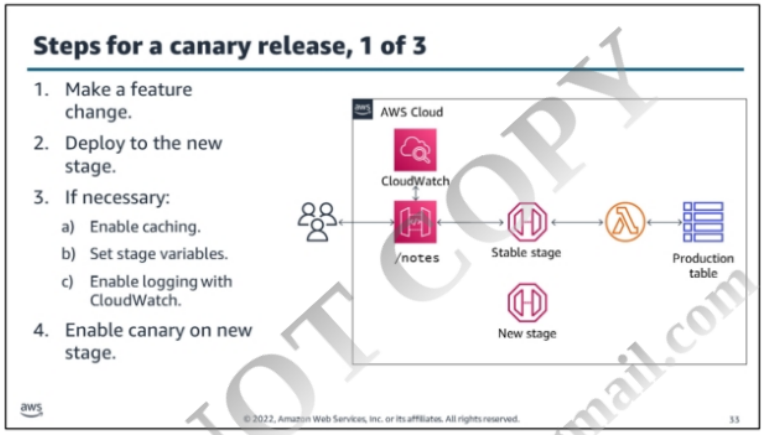
A canary release is a software development strategy in which a new version of an API or other software is slowly rolled out as a new (or testing) version of an existing application. The base version remains deployed as a production release or a non-production release for normal operations on the same stage.

The purpose of a canary deployment is to reduce the risk of deploying a new version that affects the workload. The method will incrementally deploy the new version, making it visible to new users in a slow fashion. As you gain confidence in the deployment, you will deploy it to replace the current version in its entirety.

For more information, see "Performing canary deployments for service integrations with Amazon API Gateway" in the AWS Compute Blog (https://aws.amazon.com/blogs/compute/performing-canary-deployments-for-service-integrations-with-amazon-api-gateway/).

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 10: Managing the APIs



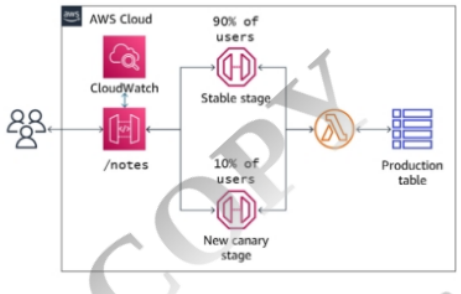
Canary deployment
The purpose of a canary deployment is to reduce the risk of deploying a new version that impacts the workload. The method will incrementally deploy the new version, making it visible to new users in a slow fashion. As you gain confidence in the deployment, you will deploy it to replace the current version in its entirety.

Code example

```
"methodSettings" : {
    "String" : {
      "metricsEnabled" : "true"
      ...
    }
},
"variables" : {
    "Var1" : "Val1"
},
"canarySettings" : {
    "percentTraffic" : "10",
    "deploymentId" : "A1b2C3",
    "useStageCache" : "False" },
```

Steps for a canary release, 2 of 3

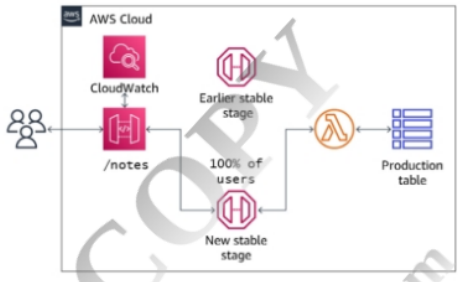
5. Set the percentage of requests to canary.
6. Re-deploy APIs to the canary-enabled stage.

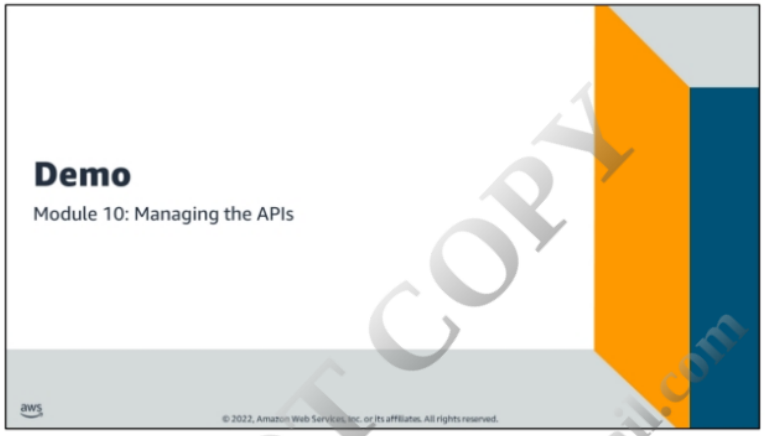


Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Steps for a canary release, 3 of 3

7. Promote canary.

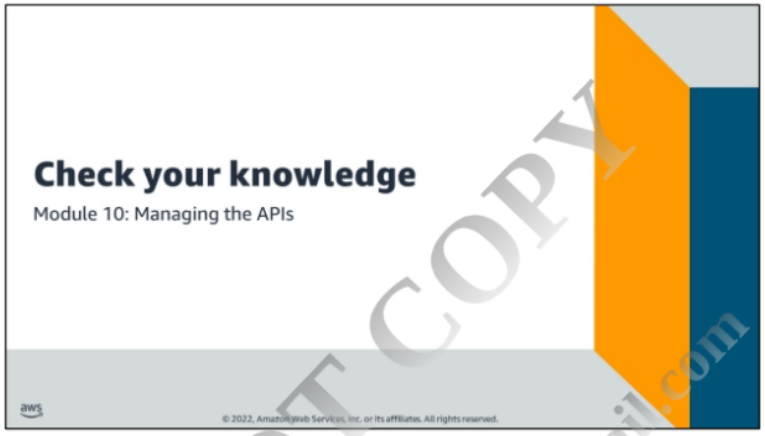


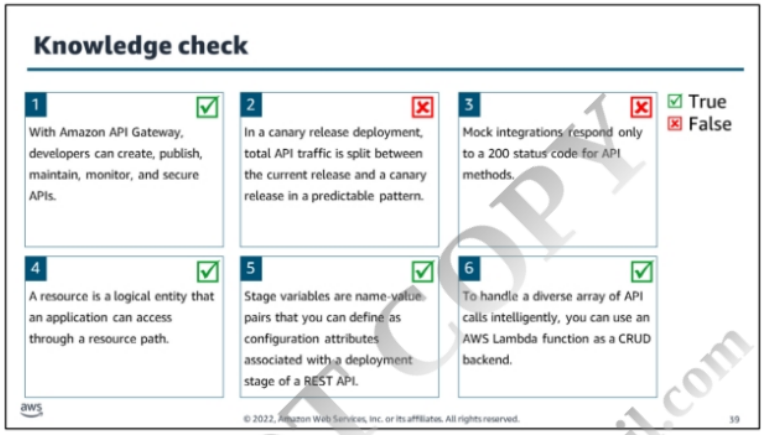


Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification

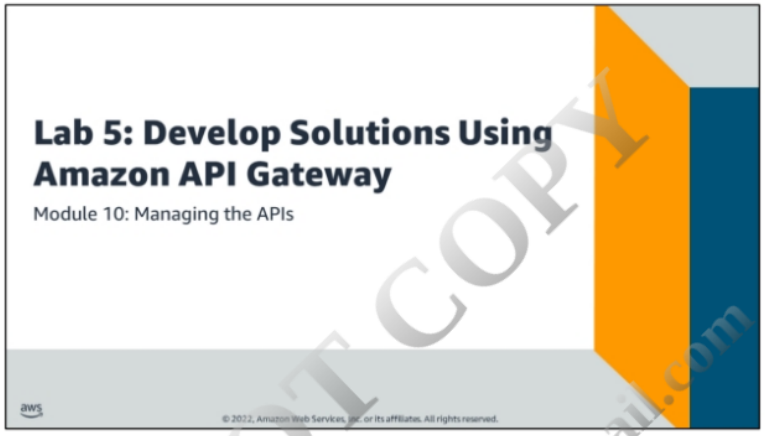
Module 10: Managing the APIs

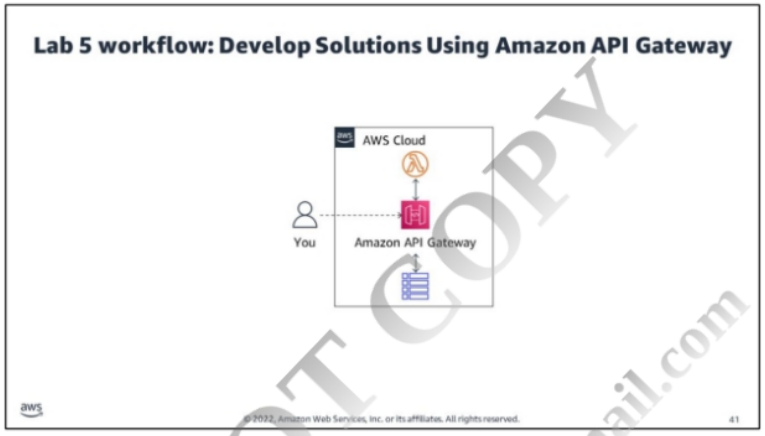




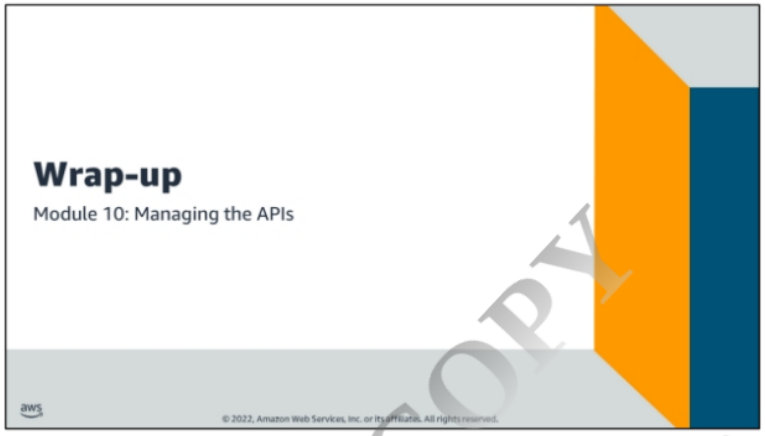
1. (True)
2. (False) In a canary release deployment, total API traffic is separated at random into a production release and a canary release with a preconfigured ratio.
3. (False) API developers can configure any status code and decide how API Gateway responds to a mock integration request.
4. (True)
5. (True)
6. (True)

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.





Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



Module summary

You are now able to:

- Describe the key components of Amazon API Gateway
- Develop API Gateway resources to integrate with AWS services
- Configure API request and response calls for your application endpoints
- Test API resources and deploy your application API endpoint
- Demonstrate creating API Gateway resources to interact with your application APIs

 43

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Thank you

Corrections, feedback, or other questions?
Contact us at https://support.aws.amazon.com/#/contacts/aws-training.
All trademarks are the property of their owners.