WSO2 API Manager 3.2.0 Developer Fundamentals

Working with Gateway

WSO2 Training

Objective

At the end of this module, attendees will get a basic idea of

- What is the API Gateway?
- API Gateway request flow
- Gateway Environments
- Gateway Caching

Understanding API Gateway

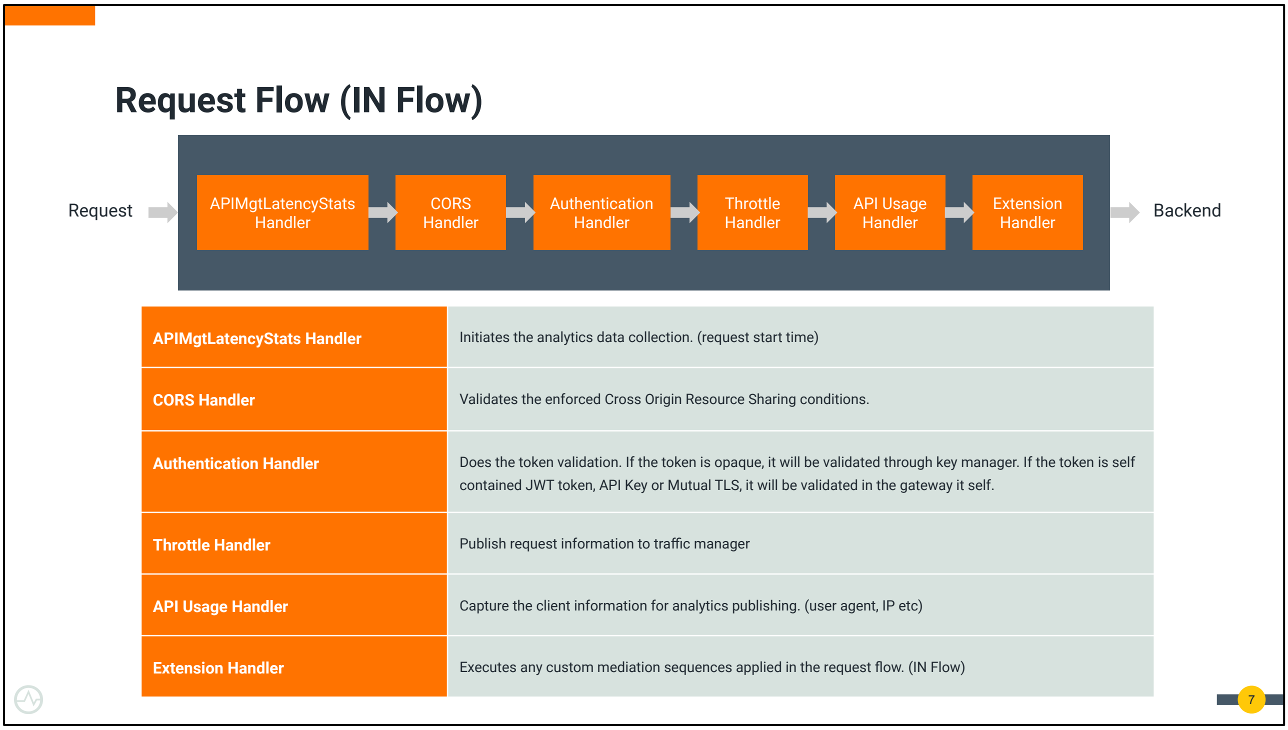
API Gateway

- API Gateway is the runtime component of API Manager and acts as a proxy for API invocations.
- When an API is published from the API publisher, it is exposed to the users via API Gateway.
- Gateway directs the API calls to the backend endpoints and enforces security, rate limiting, does stats publishing etc.
- WSO2 API Manager Gateway is developed based on the WSO2 EI which is known for its high performance.
- The default Gateway ports
    - HTTPS:- 8243
    - HTTP:- 8280

API Gateway Request/ Response Flow

Understanding Handlers

- In API Gateway, the request and response flow are controlled through a sequence of handlers.
- Handler is a synapse component which can intercept a request and response and perform specific task on it.
- Gateway use handlers for,
    - Authentication
    - Throttling
    - Analytics Publishing
    - CORS Handling
- Based on the requirement additional handlers can be implemented and integrated.

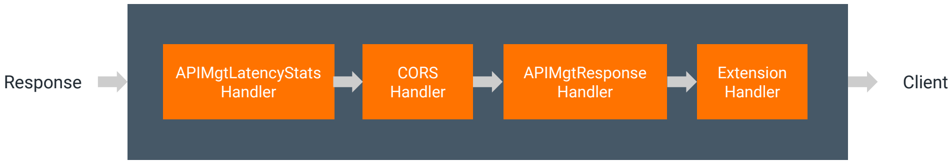


Request Flow (IN Flow)

APIMgtLatencyStats Handler	Initiates the analytics data collection. (request start time)
CORS Handler	Validates the enforced Cross Origin Resource Sharing conditions.
Authentication Handler	Does the token validation. If the token is opaque, it will be validated through key manager. If the token is self contained JWT token, API Key or Mutual TLS, it will be validated in the gateway it self.
Throttle Handler	Publish request information to traffic manager
API Usage Handler	Capture the client information for analytics publishing. (user agent, IP etc)
Extension Handler	Executes any custom mediation sequences applied in the request flow. (IN Flow)

Gateway functionalities will be discussed deeply in the Advanced Training.

Response Flow (OUT Flow)



Response

APIMgtLatencyStats Handler

CORS Handler

APIMgtResponse Handler

Extension Handler

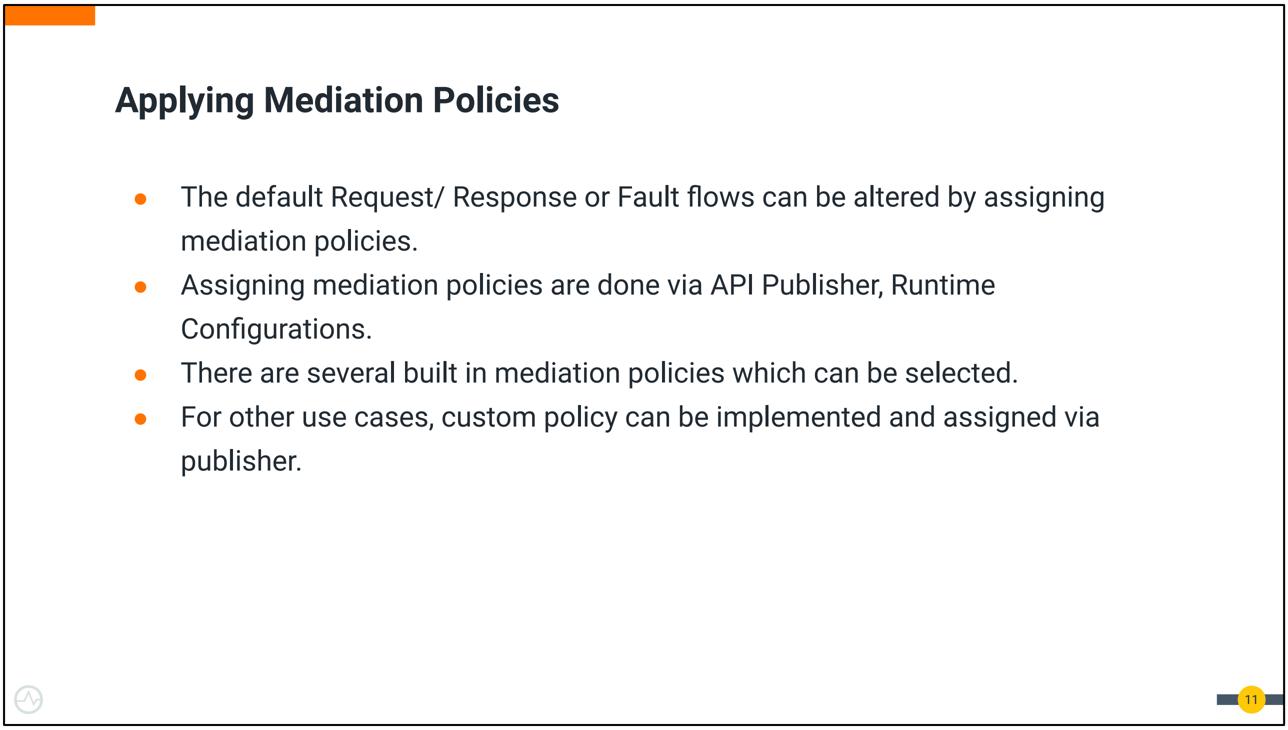
Client

APIMgtLatencyStats Handler	Collects the time the response is received
CORS Handler	Execute any CORS policies applied in response path
APIMgtResponse Handler	Publishes the captured stats to the analytics server.
Extension Handler	Executes any custom mediation sequences applied in the response flow. (OUT Flow)

Changing Default API Flows

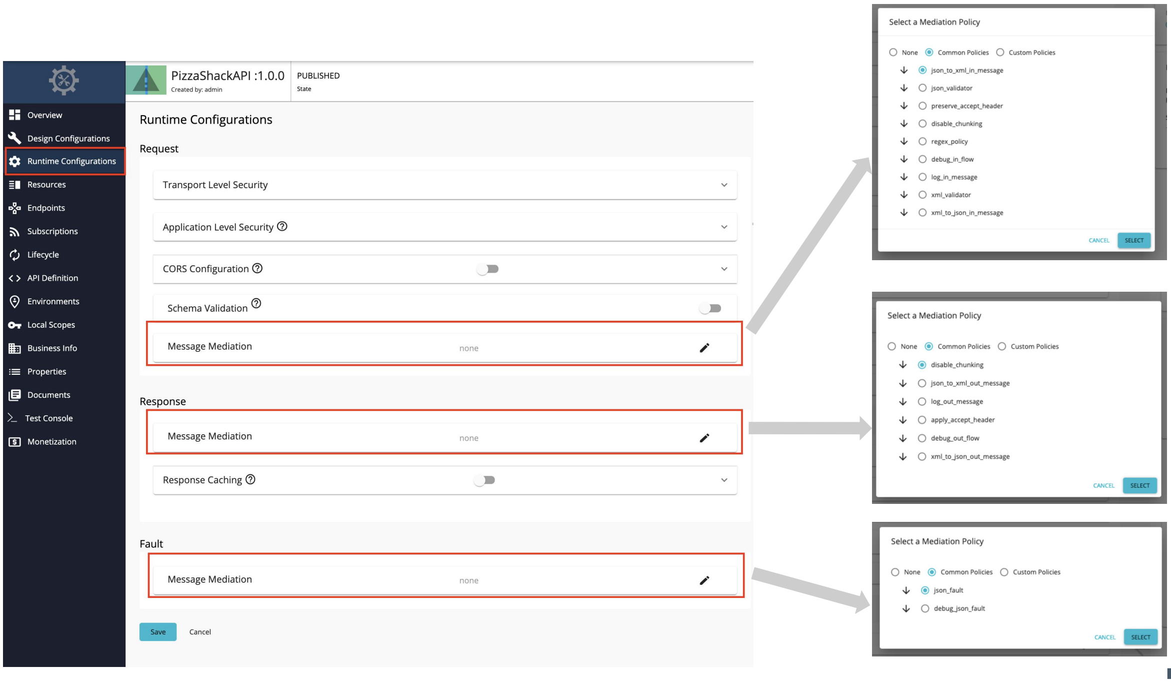
Understanding Message Mediation

- Message mediation is intercepting API request or response and performing a transformation, logging etc.
- API Gateway has three main mediation flows.
  - IN Flow : The API Request path from Client to Backend
  - OUT Flow : API response path. Backend to Client
  - Fault Flow: If any error occurred during the request/ response flow
- The default mediation flow can be altered by engaging custom mediation logic in respective mediation flows.
  - Eg: Convert JSON payload to XML
- Custom mediation logic (mediation sequences) can be implemented using WSO2 Integration Studio.



Applying Mediation Policies

- The default Request/ Response or Fault flows can be altered by assigning mediation policies.
- Assigning mediation policies are done via API Publisher, Runtime Configurations.
- There are several built in mediation policies which can be selected.
- For other use cases, custom policy can be implemented and assigned via publisher.



PizzaShackAPI :1.0.0
Created by: admin

PUBLISHED
State

Runtime Configurations

Request

Transport Level Security

Application Level Security

CORS Configuration

Schema Validation

Message Mediation none

Response

Message Mediation none

Response Caching

Fault

Message Mediation none

Save Cancel

Select a Mediation Policy

○ None ● Common Policies ○ Custom Policies

- ● json_to_xml_in_message
- ○ json_validator
- ○ preserve_accept_header
- ○ disable_chunking
- ○ regex_policy
- ○ debug_in_flow
- ○ log_in_message
- ○ xml_validator
- ○ xml_to_json_in_message

CANCEL SELECT

Select a Mediation Policy

○ None ● Common Policies ○ Custom Policies

- ● disable_chunking
- ○ json_to_xml_out_message
- ○ log_out_message
- ○ apply_accept_header
- ○ debug_out_flow
- ○ xml_to_json_out_message

CANCEL SELECT

Select a Mediation Policy

○ None ● Common Policies ○ Custom Policies

- ● json_fault
- ○ debug_json_fault

CANCEL SELECT

Gateway Environments

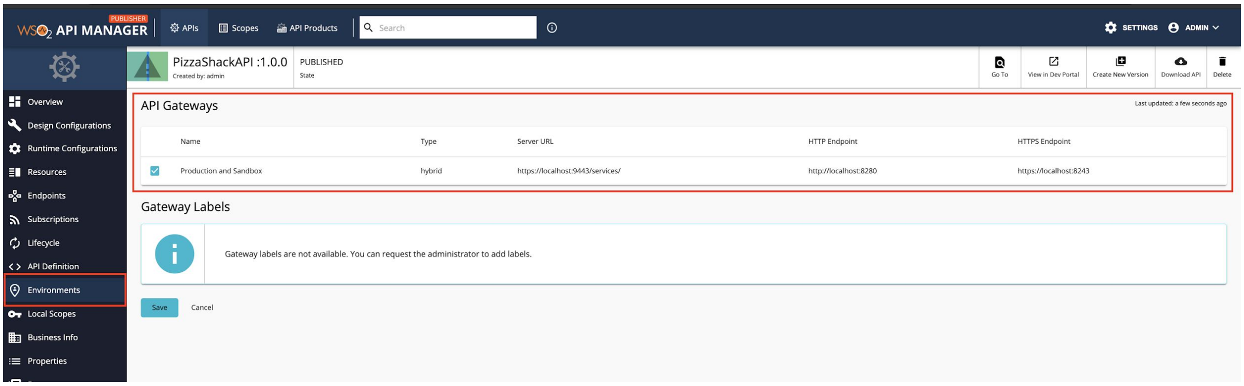
Gateway Environments

- API Gateways are considered as Environments.
- Each environment has environment name and address
- Environments are defined in <APIM_HOME>/repository/conf/deployment.toml file.

```
[[apim.gateway.environment]]
name = "Production and Sandbox"
type = "hybrid"
service_url = "https://localhost:${mgt.transport.https.port}/services/"
username= "${admin.username}"
password= "${admin.password}"
http_endpoint = "http://localhost:${http.nio.port}"
https_endpoint = "https://localhost:${https.nio.port}"
```

- In the API Publisher portal the registered Gateway environments are listed. API Publisher can select required environments when publishing the API.

Select Gateway Environment in Publisher



API Gateways

Last updated: a few seconds ago

	Name	Type	Server URL	HTTP Endpoint	HTTPS Endpoint
☑	Production and Sandbox	hybrid	https://localhost:9443/services/	http://localhost:8280	https://localhost:8243

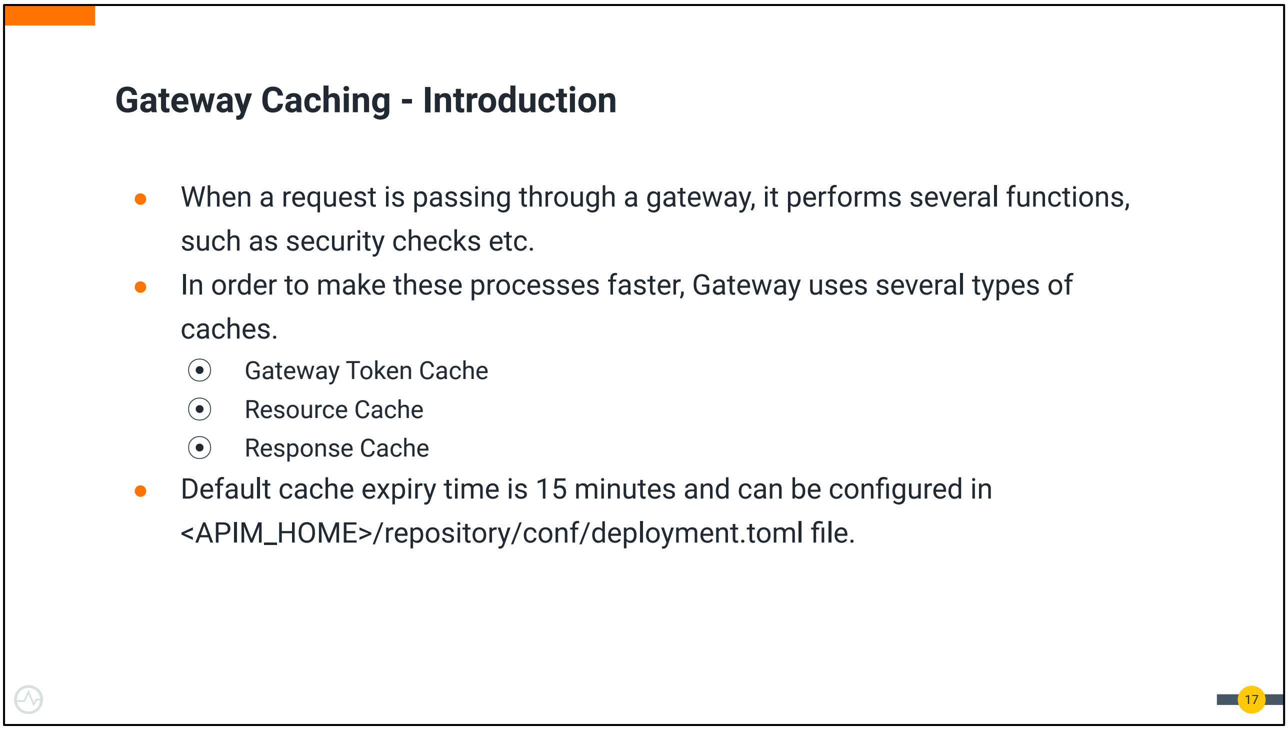
Gateway Labels

Gateway labels are not available. You can request the administrator to add labels.

- In API Publisher, open an API by selecting it.
- Go to Environments tab in left menu.
- Select the required gateway environment which the API should be published.
- Click on Save to save the changes.

If the API is in created state, go to the Lifecycle page and click on Publish button to publish the API to the selected gateway environments.

Caching



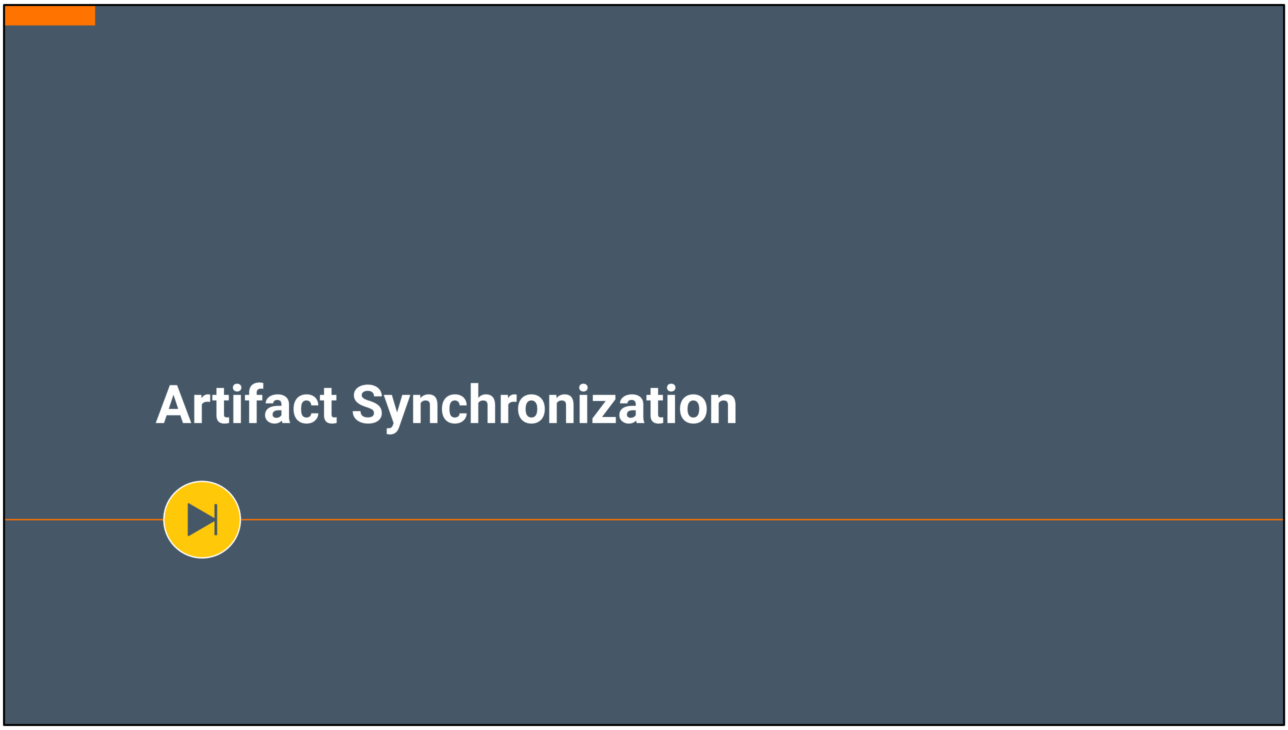
Gateway Caching - Introduction

- When a request is passing through a gateway, it performs several functions, such as security checks etc.
- In order to make these processes faster, Gateway uses several types of caches.
  - ⦿ Gateway Token Cache
  - ⦿ Resource Cache
  - ⦿ Response Cache
- Default cache expiry time is 15 minutes and can be configured in <APIM_HOME>/repository/conf/deployment.toml file.

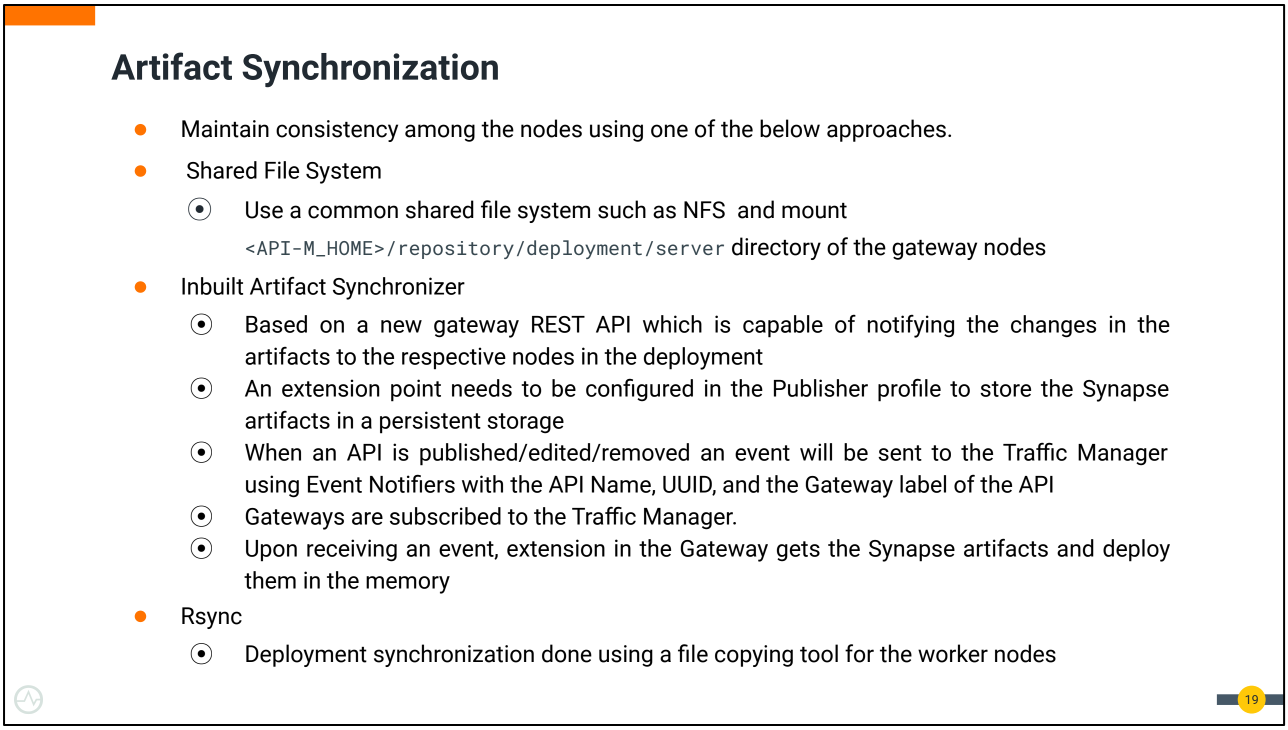
Doc link : Configure Caching

When an API call hits the API Gateway, the Gateway carries out security checks to verify if the token is valid. During these verifications, the API Gateway extracts parameters (i.e., access token, API name, and API version) that are passed on to it. As the entire load of the traffic to APIs goes through the API Gateway, this verification process needs to be fast and efficient in order to prevent overhead and delays. WSO2 API Manager uses caching for this purpose, where the validation information is cached with the token, API name, and version, and the cache is stored in either the API Gateway or the Key Manager server.

Gateway token cache - Gateway keeps a cached entry for a given token, if cached entry does not exist in the cache, it calls from the KM.
Resource cache - Cache Auth type and throttling level of the API resource call (The cache entry is identified by a cache key, which is based on the API's context, version, request path, and HTTP method)
Response cache - Back end respons can be cached in Gateway for particular API resource request, response cached time can be configure in run time configuration



Artifact Synchronization



Artifact Synchronization

- Maintain consistency among the nodes using one of the below approaches.
- Shared File System
  - Use a common shared file system such as NFS and mount `<API-M_HOME>/repository/deployment/server` directory of the gateway nodes
- Inbuilt Artifact Synchronizer
  - Based on a new gateway REST API which is capable of notifying the changes in the artifacts to the respective nodes in the deployment
  - An extension point needs to be configured in the Publisher profile to store the Synapse artifacts in a persistent storage
  - When an API is published/edited/removed an event will be sent to the Traffic Manager using Event Notifiers with the API Name, UUID, and the Gateway label of the API
  - Gateways are subscribed to the Traffic Manager.
  - Upon receiving an event, extension in the Gateway gets the Synapse artifacts and deploy them in the memory
- Rsync
  - Deployment synchronization done using a file copying tool for the worker nodes

Refer Synchronizing Artifacts in a Gateway Cluster for more details.