## **API Management**

**Key Capabilities of an API Management Platform:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Apigee** | **Benefits** |
| Simplified application development | Can create APIs quickly from existing data and services. It provides zero downtime to use those APIs, so the APIs are ready to use. Using following methods, can create API proxies in Apigee environment like importing swagger file, using Java maven plugin or create an API Proxy using Apigee UI | It speeds up the API creation and deployment. |
| Secure and scale | Manage access to APIs with security features that protect - with configurable policies like OAuth, Spike Arrest, XML/JSON threat protection, Regular Expression, Verify access token, access control, Extraction Variable, Assign message, and SAML assertions. | Apigee build trust and engagement with secure assets in your API program. It provides the protection against the hackers, bots, and other suspicious attacks |
| Reporting and monitoring | Apigee Analytics, it’s an inbuilt feature. It provides us the API performance, traffic consumption, developer engagement, API error details etc. and able to generate a custom report | Using this feature, can monitor all microservices closely |
| Improve developer productivity | Deliver an enhanced developer and community experience that accelerates API adoption, simplifies learning, and improves the business value of APIs | It speeds up development cycles, improve partner onboarding, and expand ecosystem growth. |

## **Parivartan Web/Mobile Applications Authentication?**

## Apigee is single entry point for all web and mobile users and the users are stored in LDAP. These users authenticated by OAuth2 password grant type; upon successfully authentication the user receives access token for an authorization.

## A screenshot of a cell phone Description automatically generated

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Client app invokes token endpoint for authorization access token, it passes client id, client secret, user id, password and grant type from user login screen |
| 2 | API Gateway validates client id and secret and invokes user authentication API to LDAP |
| 3 | On successful login Gateway returns access token, refresh token, and expire time with access privileges to the client app |
| 4 | For all other cases it returns invalid credentials as response |

## **Parivartan Web/Mobile Applications Authorization?**

## Apigee provides the authorization token for application access. On successfully authenticated access token user can fetch details from backend server.

## A screenshot of a cell phone Description automatically generated

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Client app invokes API endpoint with access token and request payload, Apigee GW validates and transform the API request and response using following policies (Verify Access Key, Spike Arrest, JSON/XML threat protection, Regular Expression, JavaScript, Extract Variable and Assign Message policy) |
| 2 | Verify Access Token policy – It validates the access token; on success it forwards the request to next policy |
| 3 | Spike Arrest Policy – It protects from sudden traffic spikes that can be caused by an increase in usage, buggy clients, or malicious attacks. |
| 4 | JSON / XML Threat Protection Policy – It protects from content-based attacks, it validates the structure of JSON / XML request payloads |
| 5 | Regular Expression Policy – It protects from SQL injection, all types of Code injection etc., it extracts the information from query parameters, URI path, header, form parameters variables, json and XML path variables and evaluates that against expression. If any specifies expression evaluate to true, the message is considered a threat and is rejected. |
| 6 | JavaScript Policy – It is used to validate the request payload, or it also used for response transformation |
| 7 | Extract Variable Policy – It extracts content from a request or response and sets the value of a variable to that content |
| 8 | Assign Message policy – It creates new request or response messages during the API proxy flow. |
| 9 | Response sent back to the client app |

## **Parivartan 3rd Party API Authorization and API Calls**

## Using oAuth2 client credentials grant type to authenticate 3rd party API client, on successful authentication Apigee provides the authorization token to access the business APIs

## A screenshot of a cell phone Description automatically generated

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Client app invokes token endpoint for authorization access token, it passes client id, client secret, and grant type from 3rd party APP |
| 2 | API Gateway validates client id and secret and generates the access token |
| 3 | On successful authentication Apigee GW returns access token, refresh token, and expire time with access privileges to the 3rd party APP |
| 4 | For all other cases it returns invalid credentials as response |
| 5 | Using this access token 3rd party can make business calls, like they can invoke APIs for the transaction |
| 6 | After successfully authentication of 3rd party token on Apigee the request forward to SCG which is on TAS platform to process the request |
| 7 | After successfully process the request on SCG, it forwards the response to Apigee |
| 8 | Apigee forward the response back to 3rd party API Client |

## **Apigee APIs Deployment Process**

## **Please provide these flows with respect to Apigee Intranet (on premise) deployment, as well as for Internet on (Google Cloud based deployment)?**

## Apigee Deployment Process: -

## Apigee provides following deployment process to deploy proxies from PC to Apigee environment, deploy APIs using the management UI, Command-line Scripts, or with management API’s in both cloud and on-premise environment.

|  |  |
| --- | --- |
| **Method** | **Deployment Process** |
| Management UI | Create an API proxy bundle upload into Apigee using management UICreate API proxy using reverse proxy option |
| Command-line | Java Maven ToolPythonNote: - The size limit of a proxy bundle is 15MB |
| Management APIs | Using Apigee management APIs |

## Management UI: -

## Login to Apigee Management UI.

## Click on API Proxy under Development tab to create new API proxy.

## Use reverse proxy option or upload API proxy bundle to create a new API

## Upon successfully deployment test the API proxy with postman

## Command-Line Deployment Flow: -

## Developer push/pull the API proxy code from the repository.

## Setup the CI/CD on repository deployment and for sentry checks.

## As part of Stage 1, the code will be pulled into workspace.

## In “Static Code Analysis” stage, the code will be analysed for any violations of best code practices and anti-patterns usage. If this stage is success it proceeds with build stage.

## As part of build it creates APIGEE API proxy bundle.

## In Deploy stage use APIGEE management APIS to deploy API proxy bundle.

## Once the deployment is successful, then the integration tests will be triggered. Can use Newman to do integration testes. Newman requires integration tests collection file as input. The test cases can be created easily with Postman.

## Management APIs: -

## Using management APIs implement deployment a script to create API proxy

## Upon successfully deployment test the API proxy with Postman

## **Securing Parivartan API Calls from Web and Mobile Channels**

## Note: - Replace the below sequence diagram with the old one

## A screenshot of a cell phone Description automatically generated

## **SSL Offloading in Apigee**

## Using TLS/SSL is the standard security technology for establishing an encrypted link between a web server and web client. An encrypted link ensures that all data passing between the server and the client remains private. To use TLS/SSL, a client makes a secure request to the server by using the encrypted https protocol, instead of the unencrypted http protocol.

## Apigee Supports one-way and two-way TLS in both a cloud and on-premises deployment (supported version TLS 1.2). One-way TLS enables the web client to verify the identity of the web Server. In two-way TLS, the client verifies the identity of the server followed by the server verifying the identity of the client. Implement two-way TLS to enhance security E2E and protect Parivartan data from client attacks such as client spoofing or man-in-the middle attacks.

## **TLS terminology**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| TLS Certificate | A digital file that identifies an entity in a TLS transaction. A Certificate can be used to identify the TLS server and client, depending on the TLS configuration |
| Certificate Authority (CA) | A trusted entity, such as Symantec, VeriSign or DigiCert etc., used to issue certs and to validate the authenticity of a cert. |
| Public Key | Encrypted data sent from web client to server. The public key is included in the cert. All TLS clients have a copy of the server’s public key |
| Private Key | The TLS server to decrypt data. Only the TLS server has the private key, and which is not shared with web client |
| CSR | A Certificate Signing Request (CSR) is a file generated on the TLS server based on the private key. The CSR contains the public key and other information like organization's name, location, and domain name. The CA will sign the CSR to create a TLS certificate. |
| Keystore | Contains the TLS certificate and private key used to identify the entity during TLS handshaking |
| Truststore | Contains trusted certificates on a TLS client used to validate a TLS server’s certificate presented to the client. |
| Algorithm | RSA, DSA, ECDSA, SHA, RSA with SHA, AES (Some of unbreakable algorithms Triple DES, RSA, Blowfish, Twofish and AES) |

## **One-way TLS/SSL**

## TLS/SSL handshaking for one-way authentication between a web client and web server

## A screenshot of a cell phone Description automatically generated

|  |  |
| --- | --- |
| 1 | The client issues a session request to the server |
| 2 | The server responds with a certificate, which contains its public key. It shares from the server keystore and the server also contain private key which is not shared with client |
| 3 | The client makes a request to the Certificate Authority (CA) to authenticates the certificate. |
| 4 | Upon successfully validation of the keys, the data will be encrypted and shared between the client and server. |

## **Two-way TLS/SSL**

## TLS/SSL handshaking for two-way TLS authentication between a client and server

## A screenshot of a cell phone Description automatically generated

|  |  |
| --- | --- |
| 1 | The client and server both have their own keystore and truststore to exchange the cert between the sever. |
| 2 | The **client** has a copy of the **server's** cert in its truststore. During TLS handshaking, the **client** compares the cert in its truststore to the cert send from the **server** to verify the identity of the **server.** |
| 3 | Upon successfully validation of the keys, the data will be encrypted and shared between the client and server. |

## **Key Management & Strength**

## The Advanced Encryption Standard (AES) is the algorithm trusted as the standard by the U.S. Government and numerous organizations. Although it is extremely efficient in 128-bit form, AES also uses keys of 192 and 256 bits for heavy duty encryption purposes.

## AES is largely considered impervious to all attacks, with the exception of brute force, which attempts to decipher messages using all possible combinations in the 128, 192, or 256-bit cipher. Still, security experts believe that AES will eventually be hailed the de facto standard for encrypting data in the private sector.

## **API security best practices**

## Use tokens, establish trusted identities and then control access to services and resources by using tokens assigned to those identities.

## Use encryption and signatures, encrypt data using TLS/SSL. Ensure that the right users are decrypting and modifying the data

## Identify vulnerabilities, keep up with operating system, network, drivers, and API components.

## Use quotas and throttling, place quotas and throttling to restrict the API sudden traffic and unwanted usage.

## **Apigee On-Premise Installation topology**

## The minimum hardware requirements for the installation components. The following component packages are provided by Apigee itself.

|  |  |  |  |
| --- | --- | --- | --- |
| **Installation Component** | **RAM** | **CPU** | **Minimum hard disk** |
| Cassandra | 16GB | 8-core | 250GB local storage wit SSD supporting 2000 IOPS |
| Message Processor/Router on same machine | 16GB | 8-core | 100GB |
| Message Processor (standalone) | 16GB | 8-core | 100GB |
| Router (standalone) | 16GB | 8-core | 100GB |
| Analytics – Postgres/Qpid on same server | 16GB | 8-core | 500GB-1TB, preferably with SSD backend, supporting 1000 IOPS |
| Analytics – Postgres master or standby (Standalone) | 16GB | 8-core | 500GB-1TB, preferably with SSD backend, supporting 1000 IOPS |
| Analytics – Qpid standalone | 8GB | 4-core | 30GB-50GB local storage with SSDThe default Qpid queue size is 20GB. |
| OpenLDAP/UI/Management Server | 8GB | 4-core | 60GB |
| UI/Management Server | 4GB | 2-core | 60GB |
| OpenLDAP | 4GB | 2-core | 60GB |

## **Operating System and third-party software requirements**

|  |  |
| --- | --- |
| SELinux | CentOS 7.X |
| JAVA | 1.8 |
| TLS | 1.2 (Need to check with the Apigee team, whether they support TLS 1.3) |

## **Ref link:**

## Installation Overview**:** <https://docs.apigee.com/private-cloud/v4.19.06/installation-overview>

## Topology**:** <https://docs.apigee.com/private-cloud/v4.19.06/install-edge-apigee-setup-utility>

## **MessageId**

|  |
| --- |
| Its unique ID for the request, which includes the router host name. This ID allows requests received at the router to be tracked after they are sent to the message processor. This ID is logged in Edge error logs to correlate the messageid with the errors. Add messageid to the request then forward to backend server improve the traceability of the request. |

## **Apigee Virtual Host**

## A virtual host on Edge defines the domains and Edge Router ports on which an API proxy is exposed, and, by extension, the URL that apps use to access an API proxy. A virtual host also defines whether the API proxy is accessed by using the HTTP protocol, or by the encrypted HTTPS protocol. It creates an environment level.

## For the Clearing and Exchange, we can create different environments as follow clearing & exchange

## Then the virtual host becomes, for each environment

## Clearing – [https://nseindia.clearing.com/v1/{api-path}](https://nseindia.clearing.com/v1/%7bapi-path%7d)

## Exchange – [https://nseindia.exchange.com/v1/{api-path}](https://nseindia.exchange.com/v1/%7bapi-path%7d)

## **DR Options**

## Automated routing from Apigee using target fallback server

## Manually update the DR sever details in Apigee target endpoint, then redeploy the proxy

## Manually update the DNS server details in Akamai server

## Automated routing from Akamai DNS GTM (Global Traffic Manager)

## Automated DR routing from the Apigee: -

## In this case both DC and DR should be in Active – Active mode. When DC is down, it redirects the request to the DR. Below you can find the DR routing configuration using the load balancer with health checks are configured. In this flow database always in sync between DC and DR.

## A screenshot of a cell phone Description automatically generated

## Automated DR routing from Akamai DNS GTM: -

## Using this solution, Akamai DNS exposes single endpoint to the external user to access both DC and DR application

## Using Akamai DNS GTM, it internally manages the DC – DR switch over, while DC goes down

## Ref: - <https://learn.akamai.com/en-us/webhelp/global-traffic-management/global-traffic-management-user-guide/GUID-4E6C4DD1-0617-478A-88DA-DC186E6F84FC.html>

## A screenshot of a cell phone Description automatically generated