Date: / / /

Authentication :-

Identity of an user

Authorization :-

It means whether the user has an certain rights to perform any actions (or) to access an application or not once the Authentication is successful

Bindings :- ① Http post
② Http redirect
③ Http Artifact

Http post :-

It will transmit SAML response & request message within an HTML form.

* used for trusted session b/n IDP & SP &

Http Redirect :-

Mostly used to redirect SAML authen request from SP to IDP

* It will send GET params responses

HTTP Artifact :-

It will send dummy msg and then sends an response

* It must used when requestor & responder has direct communication.

Seperate binding SOAP (Simple object Access Protocol)

Date: / / /

Onboarding (or) Integrating an Application with SAML.

1. Entity ID
2. ACS URL
3. User attributes which needs to be pass to sp.
4. certificate (Not mandatory) [only used when SLO]
5. Groups (or) AD Groups which we have to allow the access to an application.

SSO - Single Sign on

It is an authentication process that allows an users to securly authenticate to multiple apps & websites using only one set of credentials.

Is SAML uses SSO?

SAML enables SSO, which implies that user can log in to multiple apps with only one set credentials and same can be reused to log in to other sp.

SLO (Single Logout)

(App session & okta session)

→ It will kills current session and redirects sends to URL to which SLO is set

→

MATRIKAS

Date: / / /

What is SAML & how does it works?

SAML - security Assertion Markup language

It is an open standard used for authentication. It is in XML format (Extensible Markup Language.

* Web applications uses SAML to transfer authentication data b/n two parties. b/n Identity provider (IdP) & Service provider (SP)

Works:-

SAML Works by exchanging user information such as logins, authentication state, identifiers and other relavant attributes b/n the identity and service provider.

* it simplifies and secures the authentication process as the user only needs to log in once with a single set of authentication credentials

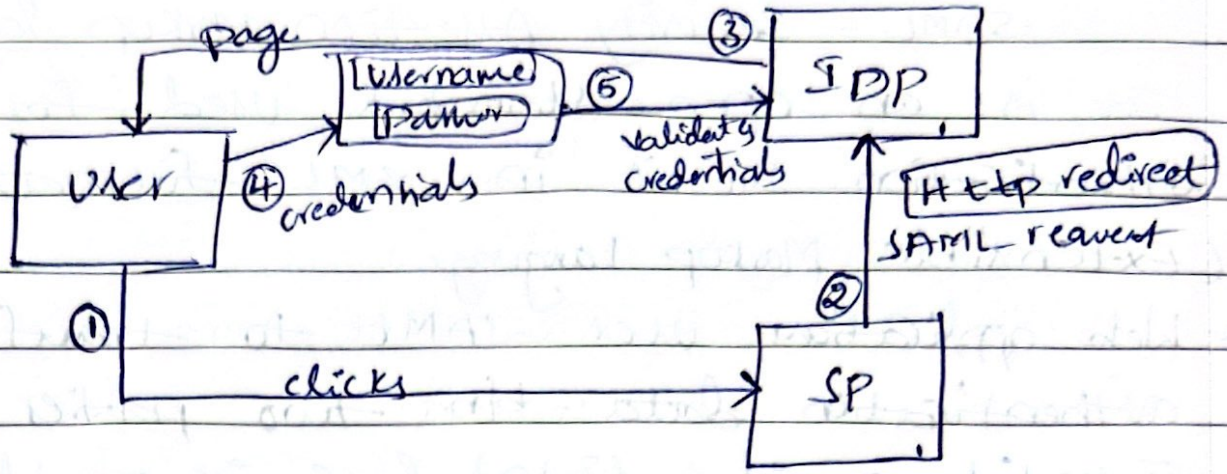
Benefits:-

* It improves the user experience as we only need to sign in once to access multiple Web apps.
* It speed up the authentication process
* It saves our time by decreasing help desk calls for password resets
* It increased security solutions as MFA
* better user experience.

MATRIKAS

Date: / / /

Sp to IDP flow :-



1. User hits the Sp url on browser
2. Sp will redirect to IDP by SAML authentication request if session is not found
3. IDP will through an login page to user through browser.
4. Login page opens with username & password
5. User has provide the credentials
6. The credentials will be validated in IDP, if the credentials which are given are correct then Authentication is success
* Session will generate and checks for an authorization.
7. Then SAML assertion has been posted to ACS url (Assertion consumer service) upon successfull authentication & Authorization.

MATRIKAS

Date: / / /

IDP to SP flow:-

1. User clicks on IDP (SSO) URL
2. IDP will through login page with an username & password.
3. User has to provide credentials.
4. credentials are validated in IDP & If given credentials are correct then Authentication is succes & session will generate.
5. SAML Assertion has been posted to ACS upon succes full Authentication & Authorization

Assertion contains:-

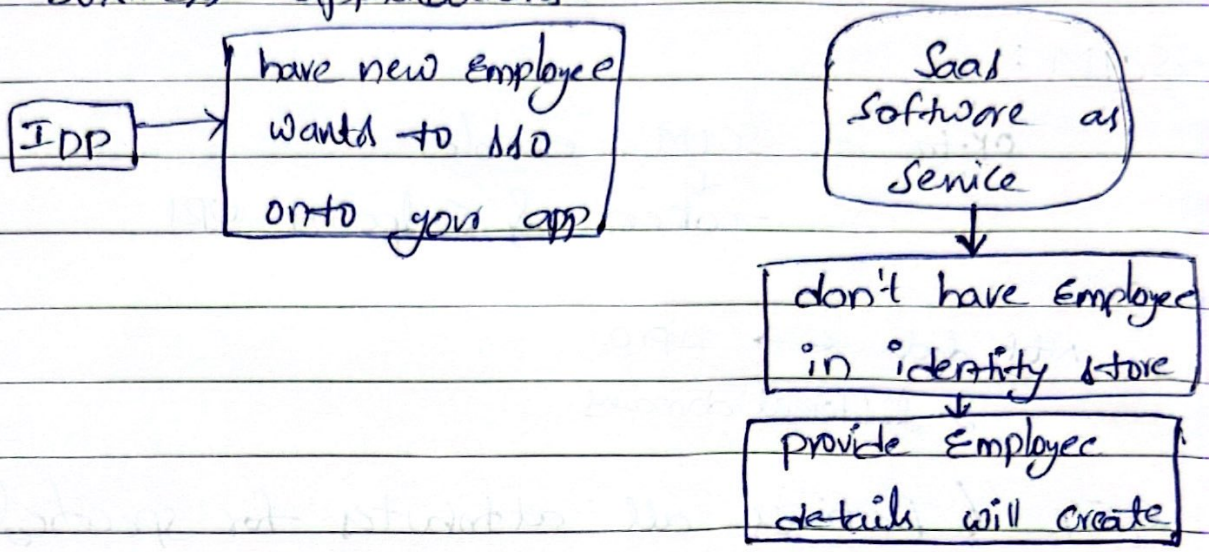
① User Attributes (F.Name, L.Name, Email, address etc).
② Entity ID
③ IDP url, ACS URL
④ Certificate.

MATRIKAS

Date: / / /

provisioning :-

provisioning technology creates, modifies, disables and deletes user accounts and their profiles across IT infrastructure and business applications.



3 kinds of user provisioning

1. JIT (Just in Time) provisioning
2. outbound provisioning
3. SCIM (Simple cloud identity management)
   (system for cross domain identity management)

JIT provisioning (SP side)

At the time of SAML assertion user will be created using assertion information.

* have to map required attributes into repository.

database		
LDAP AD	First name ↔	user first name
	Email ↔	user email

MATRIKAS

Date: / / /

Outbound provisioning :- (IDP side)

* They maintain separate database internally
* keeps user data synchronized b/n the organisation and hosted Saas application.

SCIM :-

okta - SCIM enable
+
Token & Endpoint URI

Ex

Ntt data ↔ Wipro
2 different domains

GET (Retrives all attributes for specified resource)
POST (Sends user attributes to create a new role)
PUT (updates (or) replaces an existing resources)
DELETE (Deletes or disables a resource)

Date: / / /

OAuth	OpenId
1) It is open standard authorization protocol	1) It is used for ~~authorization~~ authentication on top of OAuth
2) passes access token	2) passes access token & ID token

JWT token - JSON Web Token (user attributes or user claims (First name, email, last name, etc))

ID token - user Authentication details

Access token - user Authorization details (what type of role)

SAML	OAuth
1) uses XML to pass msgs	1) uses Json
2) SAML is geared towards clients security	2) It provides simpler mobile experience
3) It drops a session cookie in a browser that allows a user to access certain page	3) It uses API calls extensively, which is why mobile apps, modern Web apps, game console etc,
4) used for short-lived work days	4) It gives better experiences to users.
	5) Access token has less lifetime
	6) If we want to get new token using refresh

Date / / /

OAuth :-

* It is an open standard authorization protocol (or) framework that enables a third party apps to obtain limited access to an Http services.
* oAuth doesn't share password data but instead uses authorization tokens to prove an identity b/n consumers & sp

oAuth Roles :-

Resource owner :- An Entity that can grant access to protected resource [End user]

Resource server :- Server hosting the protected resources. This is an API want to access

client :- Application requesting access to a protected resource on behalf of the resource owner.

Authorization server :- Server that authenticates the resource owner and issues access token after getting proper authorization

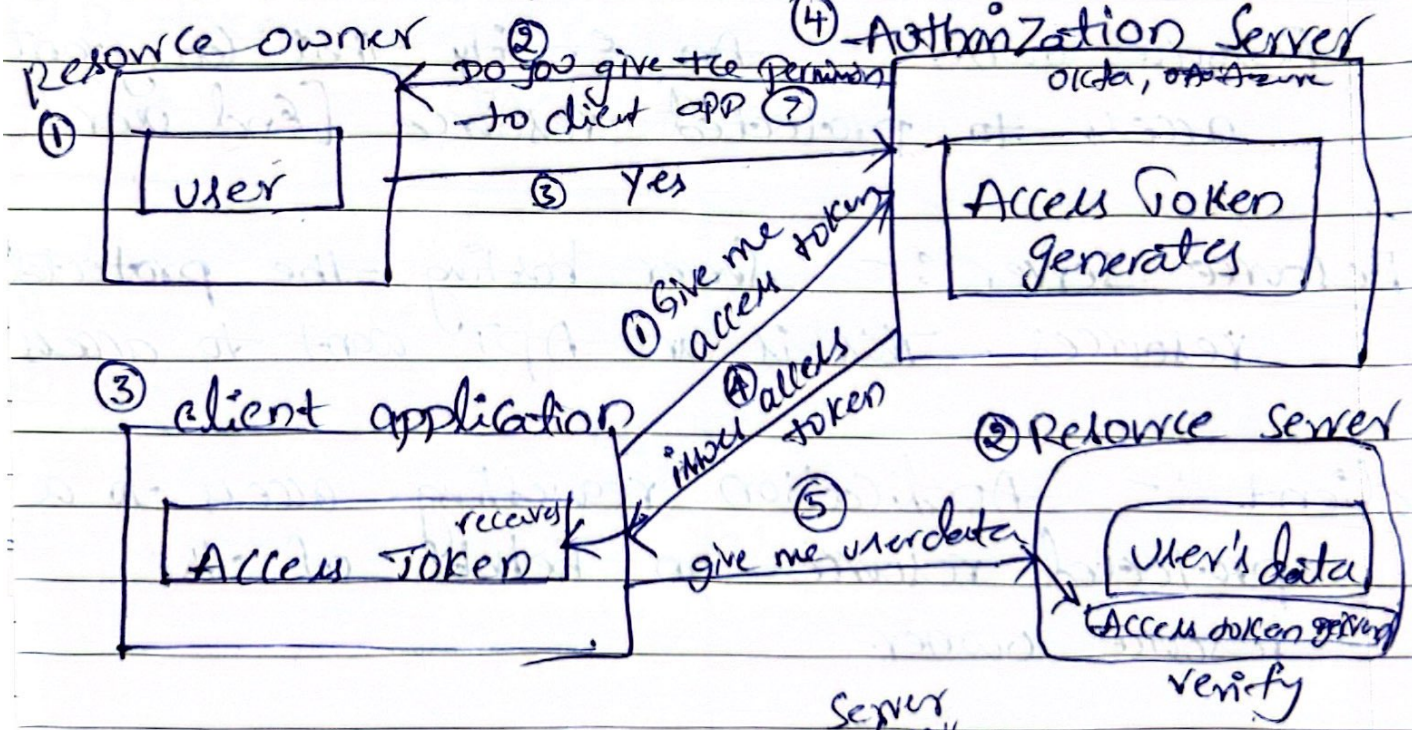
MATRIKAS

Date: / / /

Authorization Grant types :-

We can choose grant types based on the application (which is suitable for app).

1) Authorization code flow
2) Implicit flow
3) Resource owner password credentials flow
4) client credentials flow
5) Refresh Token flow.



Server resource extracts the access token from requests

MATRIKAS

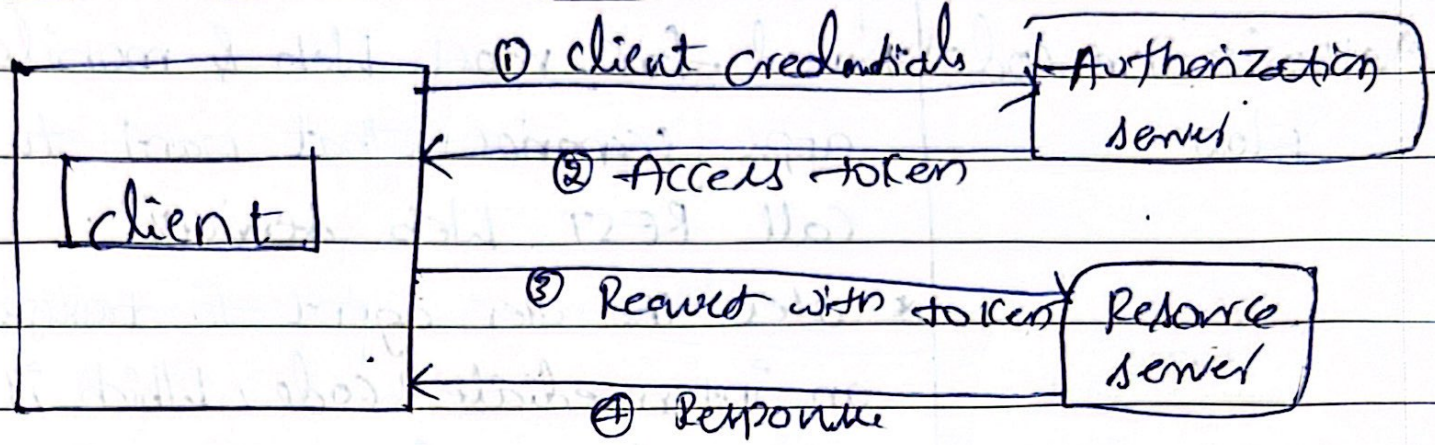
Date: / / /

Authorization code Flow	* used for most Web & mobile apps scenarios, that want to call REST Web services. * uses the user agent to transport an intermediate code, which is then exchanges for oAuth token
Implicit code Flow	* Where client is not able to safely hide the client secret (ex - client side Javascript app) * uses user agent to transport tokens
Resource owner password credentials Flow	* When app need to control the own login Page login form (ex Native mobile) * Exchanges a username/password combination for token.
client credentials Flow	* When an user is not involved Access token only required for a service to call a REST web service * Exchanges client credentials for an access token.

MATRIKAS

It will not ask for user credentials directly Date: login / /

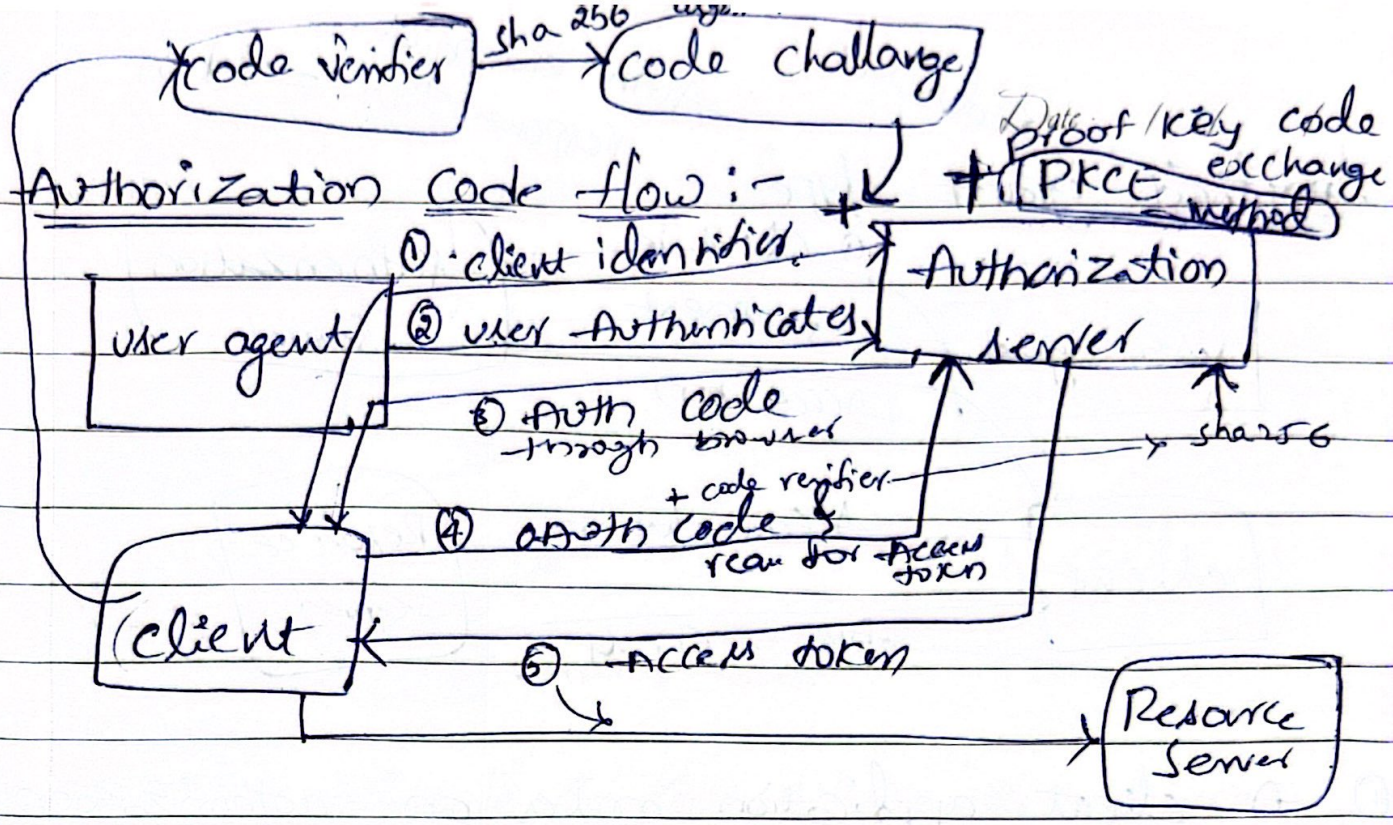
Client credentials Flow :-



① Server to server communication.
② Machine to Machine communication

* client provides client credentials (client ID) to okta (Authorization server)
* okta validates the client credentials and generates a access token.
* It will also evaluate any required policies around the token space.
* client provides the token in a HTTP headers when making a request to the resource server.
* Resource server validates the token and process the request if valid.

client access token URL



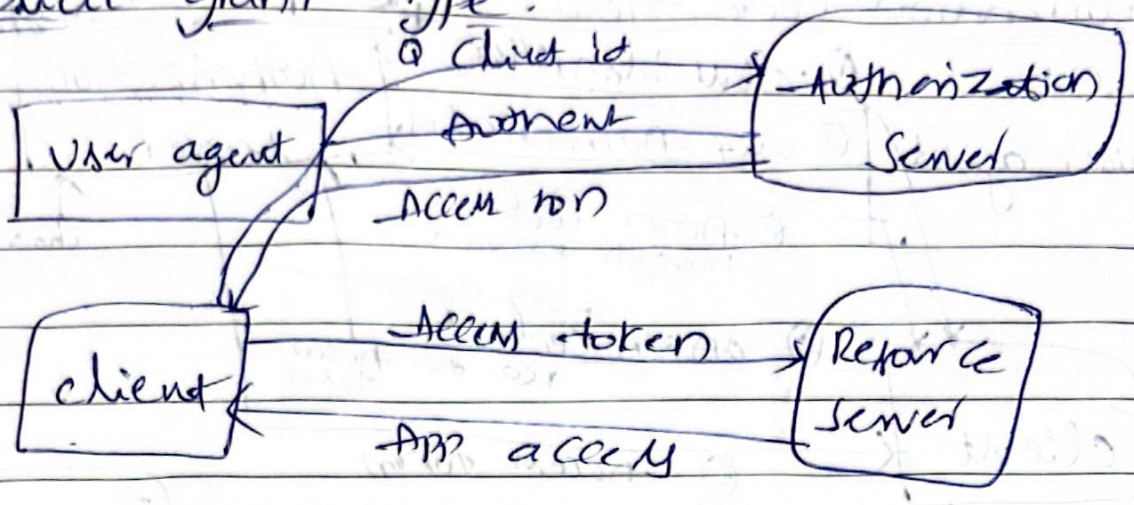
* It obtain both access token & refresh token.
* A client application makes an authorization request + code challange to an auth server
* Auth server sends Auth code to client through browser (user agent)
* client sends the Auth code + code verifier to Auth server & requests for access token
* Auth server validates Auth code & gives access token
* sends access token to Resource server and access to app.

Code verifier :- It a random value generated by client

* User request the authorization code to authorization server via user agent

redirect Auth endpoint
response type = token

Implicit grant type :-



① A client application makes an authorization request to an Authorization Server.
② Authorization server provides Access token to the user via browser
③ client provides access token to resource server and gives the app.

1) client application makes an a authorization request redirecting to the user via browser
2) the redirect contains attributes
   1) authorization code endpoint
   2) clientid
   3) response-type = code (or token)
   4) scope
   5) redirect url - application redirect URI

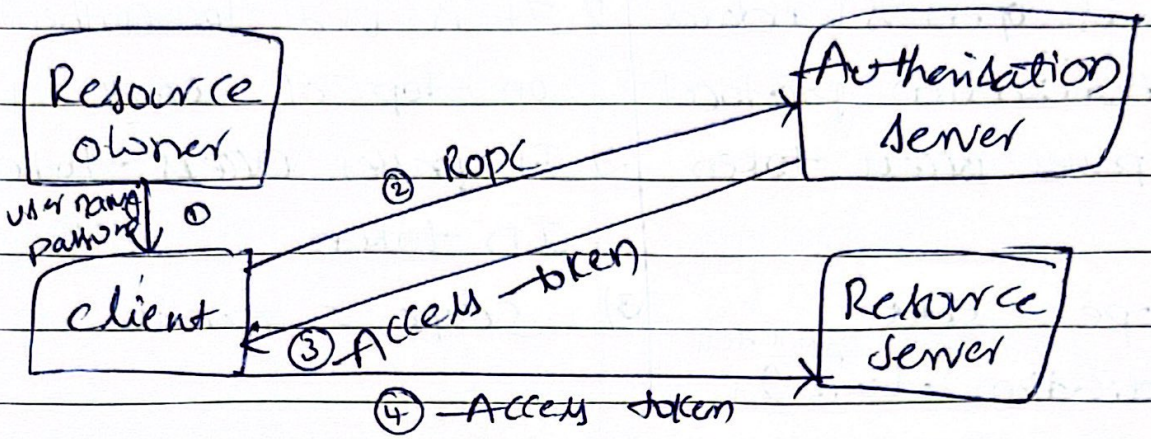
Auth code

auth code post - token endpoint URL
auth code

MATRIKAS

Date: / / /

Resource owner password credentials :-



* This is suitable in case where the resource owner has a trust relationship with client
* It can be used only when other flows are not available there uses details (user

1) Resource owner provides the client with its user name & password
2) client request as access token by using client credentials.
3) If client authenticates with it then it it issues the access token.

Date: / / /

OAuth	OpenID
1) It is open standard authorization protocol	1) It is used for authentication on top of OAuth.
2) It passes access token	2) It passes access token & ID token.
3) scope = edit (application team) (will tell)	3) scope = openid

ID Token	Access token
1) Assume the user is authenticated	* Call on API
2) Get user profile data	* check if the client is allowed to access something
	* Inspect its content on the server side

If decoded

ID

Header { alg : RS256
typ : JWT
}

Payload {
iss : http://
sub : auth
audien : 123
exp : 15/1
iat :

Access

Header { alg : Rs256
typ : JWT
kid : MDCs. . .
}

Payload {
iss : http:
sub : Mo4. .
aud : http:
iat : 16 21
exp :

Date: / / /

name : John Doe
given_name : John
family_name : Doe

Signature:-
{
RSA SHA256 (
base encod (Header)
" " (Payload)
}secret

scope : create:term,
update:term
gty : authori_code

Signature :
{
RSA SHA 256
base encode (Header)
" " (Payload)
}secret

(Key id)
Kid: optional in header claim.
* which holds a key identifier
* particularly useful when you have multiple keys to sign the tokens & you need to look up the right one to verify the signature

Date: / / /

Life cycle management :-

* Every object has its own life cyle