3SCALE

OPENIDCONNECT

CUSTOMER'S MOTIVATIONS FOR CONTAINERISATION...

- Breaking down our monolith's will give us faster time to market.
- Greater flexibility in tactically changing our applications.
- Fully automated testing - much better QA and less reliance on manual testing.
- Free us from 1 technology - opportunity to innovate with new technologies.
- Nothing stands still, and nothing stays the same - if we do not do this then our competitors certainly will!

THE STUFF OF APPDEV NIGHTMARES

Data Breaches / Data stolen

- LinkedIn (2012) - 117, 000, 000 records stolen.
- Sony - 77, 000, 000 PSN records stolen.
- DropBox (2012) - 68, 700, 000.
- UK - NHS, 8,300,000.
- US Military, 76,000,000 .
- MYSpace, 164,000,000
- Ebay, 145,000,000
- There’s plenty, plenty more….

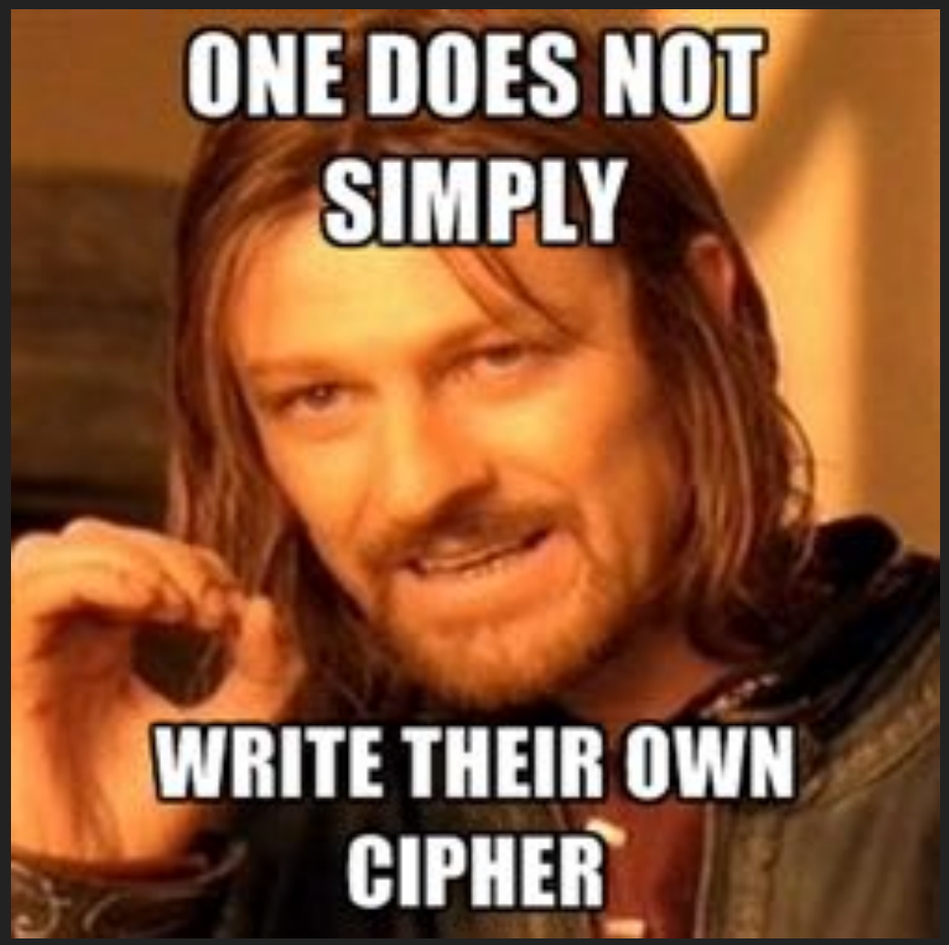
. . .AND OF COURSE IT HAS TO BE SECURE.

...This raises many questions:

- Independent services are no longer bound to one monolith ie. just 1 security context on one cluster.
- So in microservices - would each service have to store identity?
- Services owning personal data is a BIG risk if not done properly.

. . . MOST DEV'S 'DON'T DO' OR UNDERSTAND SECURITY.

- The bar for appdev security has been raised substantially.
- Most devs. don't really 'do' security - it's a specialism.
- Reputation is lost very quickly if a security leak occurs.
- The hand rolled solutions of the past are not viable going forwards.
- Are you still sure you want to store 'identity' in your App?



. . . BUT THIS WAS A PROBLEM WE FIXED A LONG TIME AGO.

- Most people do not keep their savings under their mattress for the same reasons.
- They store them in a secure environment - ie. the bank.
- Then use a separate means to authorise / authenticate payments.

. . . OAUTH / OPENIDCONNECT USES A TRUSTED IDENTITY AUTHORITY.

- Identity is abstracted away to a single ‘fortress’.
- Tokens are issued as a means of allowing access to services and resources.
- So now your application’s services do not need to store sensitive data!

OAUTH2 FROM 20,000 FT.

Federated identity - you can login and access one ‘account’ (security context) using another.

Delegated authority -one service can request access to resources on another service on the behalf of the user .

OPEN ID CONNECT

- Provides an identity framework built on to OAuth2 flows.
- Adds authorisation to OAuth2 delegated authority.
- Implemented using rest endpoints.
- Adds the /user_details endpoint which accepts an access_token and returns id_token - which represents the resource owner,
- Uses signed Jason Web Tokens - JWT (pronounced ‘jot’).
- Like SAML - but not just webpage centric, and much easier to implement.

OAUTH2 GRANT TYPES

The code grants represent different use cases:

- Authorization Code - Server side services.
- Implicit Code - Front End Javascript Clients
- Client Credentials - Application owns the resource
- Resource Owner - Application directly passes user / pass.

ID_TOKEN

- Provides identity information to the application from the Authority Server (RH - SSO)
- Base 64 encrypted - easy to work with.

OAUTH2 CODE GRANTS

AUTHORISATION CODE

(. . . FOR SERVER SIDE APPLICATION USE CASES)

OAuth2 Specification

AN ORIENTATION...

User

RH-SSO

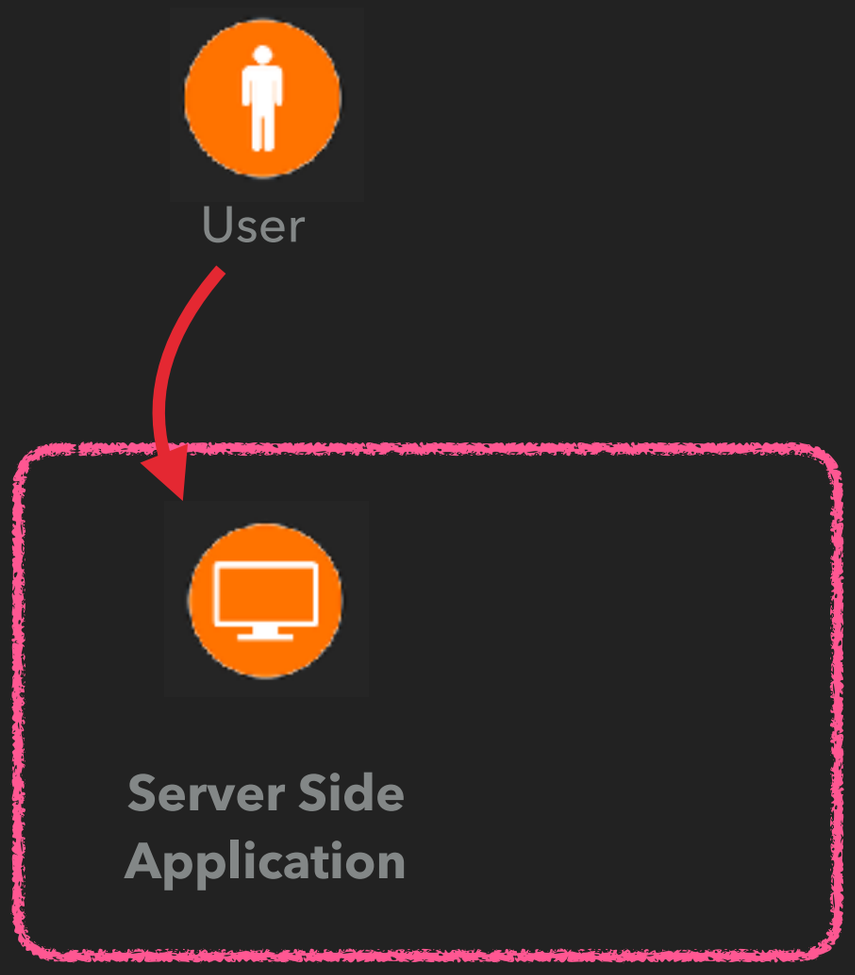
Server Side Application

3Scale /buystuff

/buystuff

Service

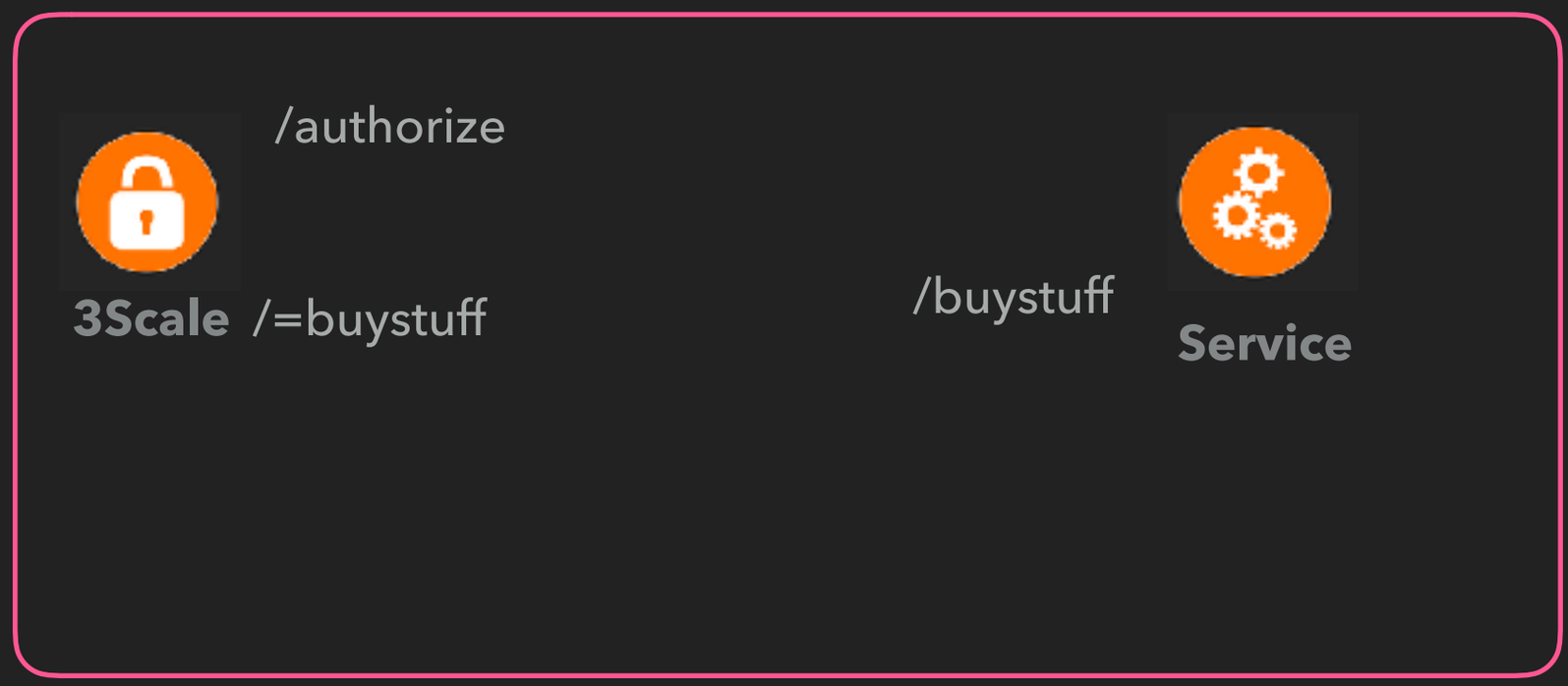
SERVER SIDE APPLICATION.



The client application can safely abstract data safely away from the user.

Therefore it can be trusted.

This is the most secure OAuth2 grant type (flow).



#4 STEPS TO IMPLEMENTATION.

STEP 1 – REGISTER YOUR APPLICATION WITH AN OAUTH PROVIDER

User

RH-SSO

client_id=askdjfkjk
secret=92kdksk3kd3k3d9w90
Auth Redirect = https://server.app.com/redirect
Authorisation Endpoint = https://redhat-sso.com

Server Side Application

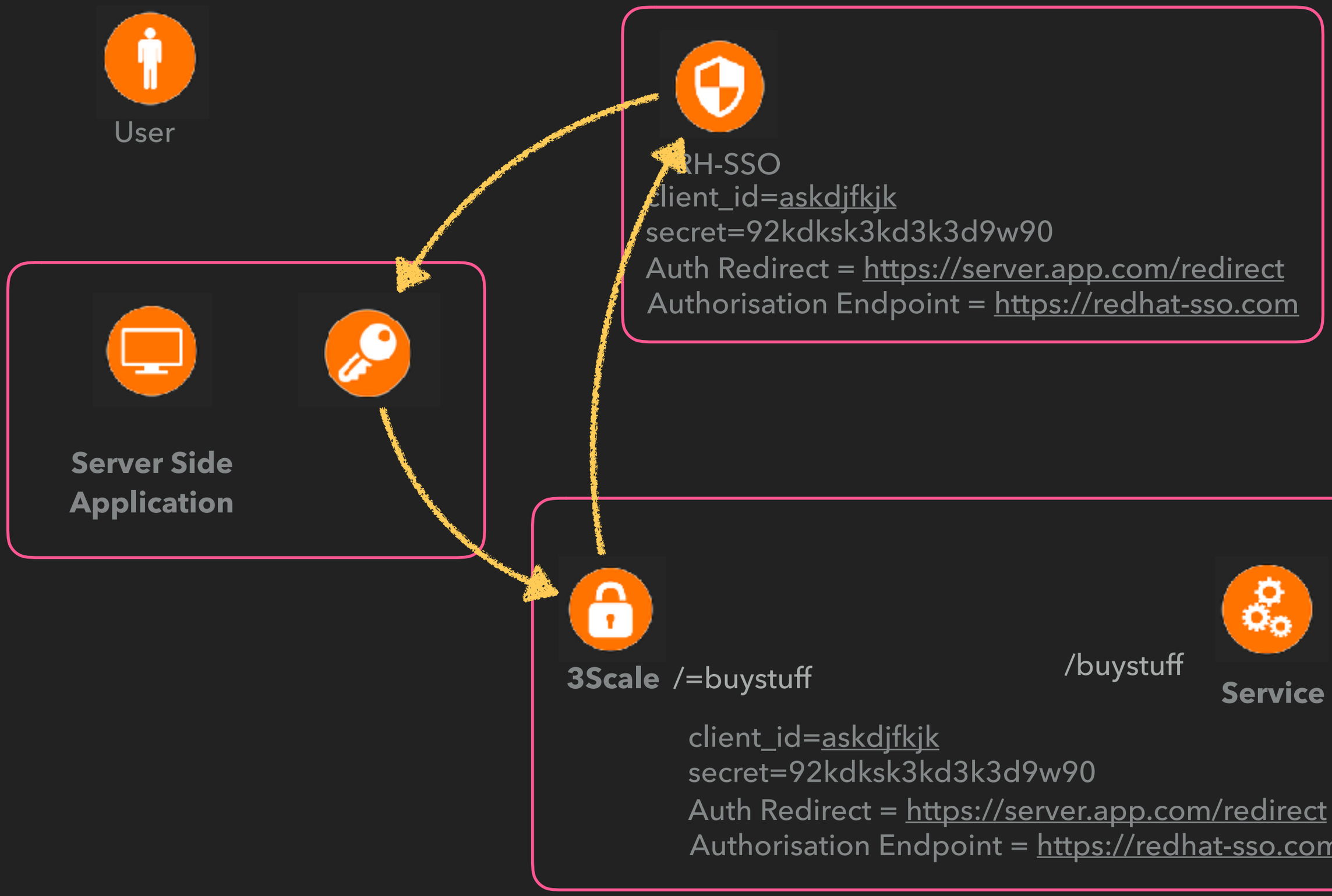
3Scale /=buystuff

/buystuff

Service

client_id=askdjfkjk
secret=92kdksk3kd3k3d9w90
Auth Redirect = https://server.app.com/redirect
Authorisation Endpoint = https://redhat-sso.com

STEP 2 – GET AN ACCESS TOKEN



STEP 3 – USE YOUR TOKEN.

User

RH-SSO
client_id=askdjfkjk
secret=92kdksk3kd3k3d9w90
Auth Redirect = https://server.app.com/redirect
Authorisation Endpoint = https://redhat-sso.com

Server Side Application

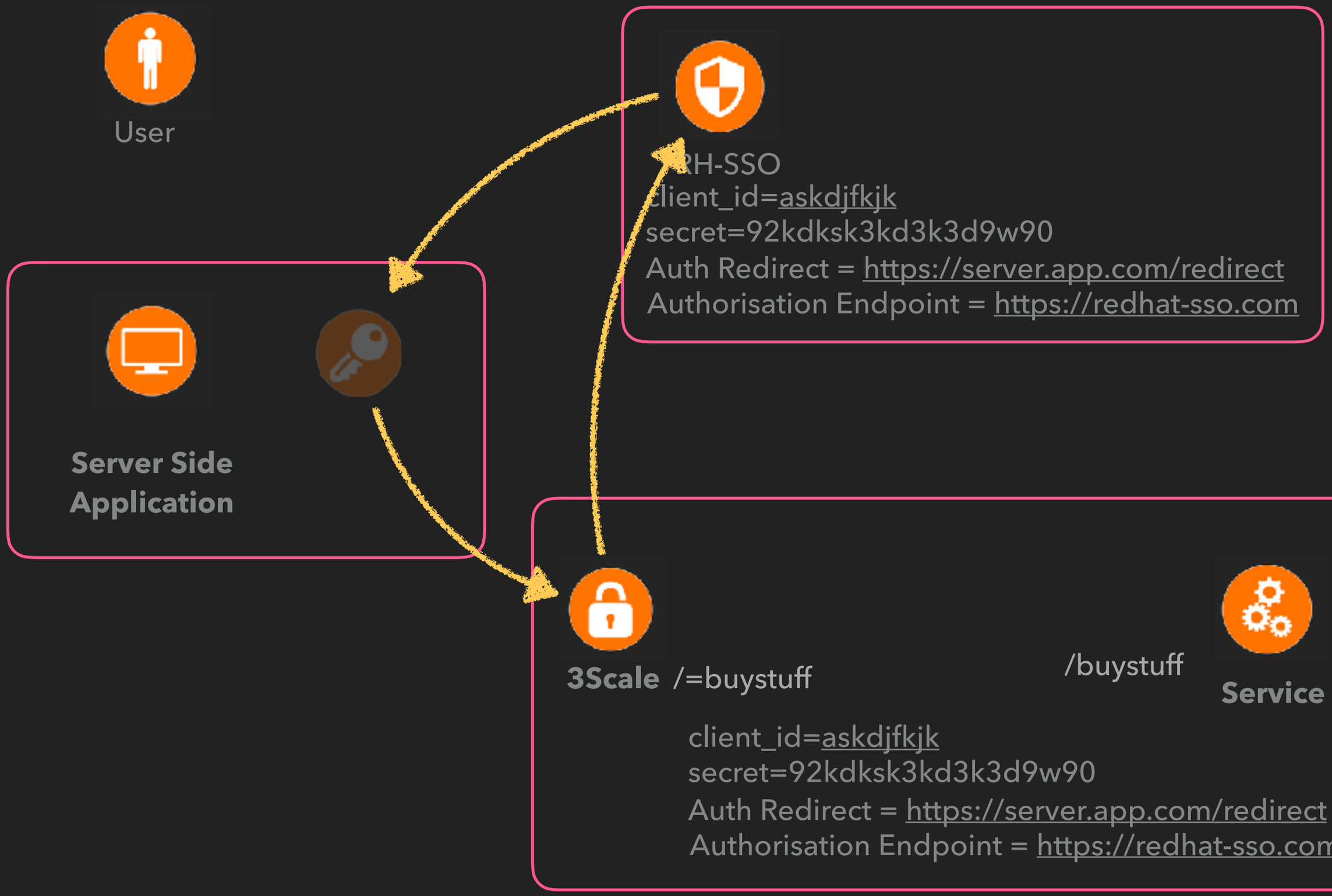
3Scale /=buystuff

/buystuff

Service

client_id=askdjfkjk
secret=92kdksk3kd3k3d9w90
Auth Redirect = https://server.app.com/redirect
Authorisation Endpoint = https://redhat-sso.com

STEP 4 – REFRESH AN EXPIRED TOKEN



THE 'BACK CHANNEL' IS INHERENTLY SECURE!

BACK CHANNEL – USER CANNOT INTERFERE WITH THIS!

User

RH-SSO
client_id=askdjfkjk
secret=92kdksk3kd3k3d9w90
Auth Redirect = https://server.app.com/redirect
Authorisation Endpoint = https://redhat-sso.com

url=http://gateway.com

Server Side
Application

3Scale /=buystuff

/buystuff

Service

client_id=askdjfkjk
secret=92kdksk3kd3k3d9w90
Auth Redirect = https://server.app.com/redirect
Authorisation Endpoint = https://redhat-sso.com

BACK CHANNEL – USER CANNOT INTERFERE WITH THIS!

User

RH-SSO
client_id=askdjfkjk
secret=92kdksk3kd3k3d9w90
Auth Redirect = https://server.app.com/redirect
Authorisation Endpoint = https://redhat-sso.com

url=http://gateway.com

Server Side Application

3Scale /=buystuff

client_id=askdjfkjk
secret=92kdksk3kd3k3d9w90
Auth Redirect = https://server.app.com/redirect
Authorisation Endpoint = https://redhat-sso.com

/buystuff

Service

BACK CHANNEL – USER CANNOT INTERFERE WITH THIS!

User

RH-SSO
client_id=askdjfkjk
secret=92kdksk3kd3k3d9w90
Auth Redirect = http://server.app.com/callback
Authorisation Endpoint = https://redhat-sso.com

url=http://gateway.com

Server Side
Application

3Scale /=buystuff

/buystuff

Service

client_id=askdjfkjk
secret=92kdksk3kd3k3d9w90
Auth Redirect = https://server.app.com/redirect
Authorisation Endpoint = https://redhat-sso.com

AUTHORISTION CODE GRANT IN ACTION.

#1 – USER HITS SERVER SIDE APPLICATION

User

Server Side Application

/authorize

/token

RH-SSO

client_id=askdjfkjk

secret=92kdksk3kd3k3d9w90

Auth Redirect = https://server.app.com/callback

Authorisation Endpoint = https://redhat-sso.com

3Scale /=buystuff

/buystuff

Service

client_id=askdjfkjk

secret=92kdksk3kd3k3d9w90

Auth Redirect = https://server.app.com/callback

Authorisation Endpoint = https://redhat-sso.com

#2 – CALLS THE GATEWAY – AUTHORISE ENDPOINT

User

Server Side Application

/authorize

/token

RH-SSO
client_id=askdjfkjk
secret=92kdksk3kd3k3d9w90
Auth Redirect = https://server.app.com/callback
Authorisation Endpoint = https://redhat-sso.com

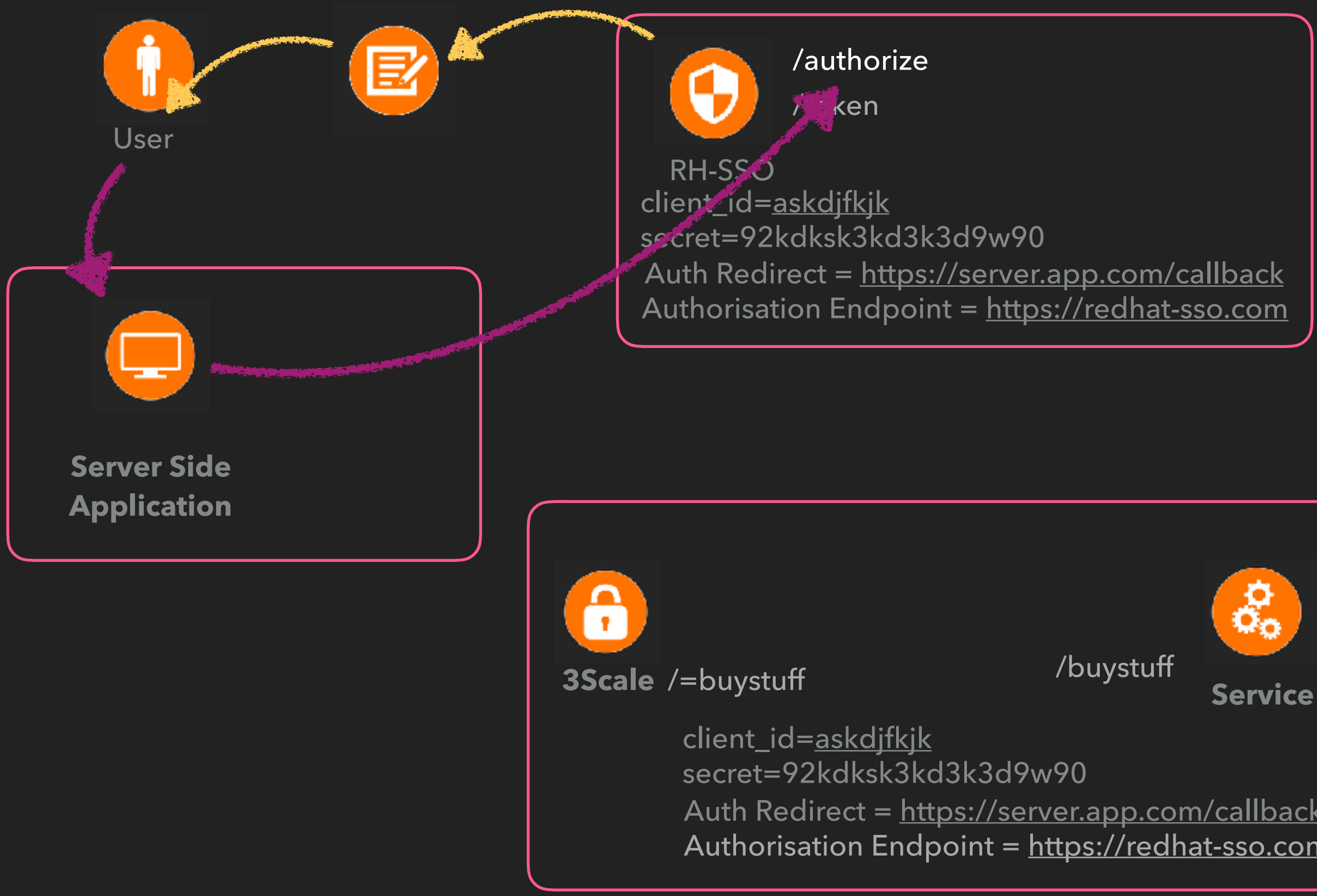
3Scale /=buystuff

/buystuff

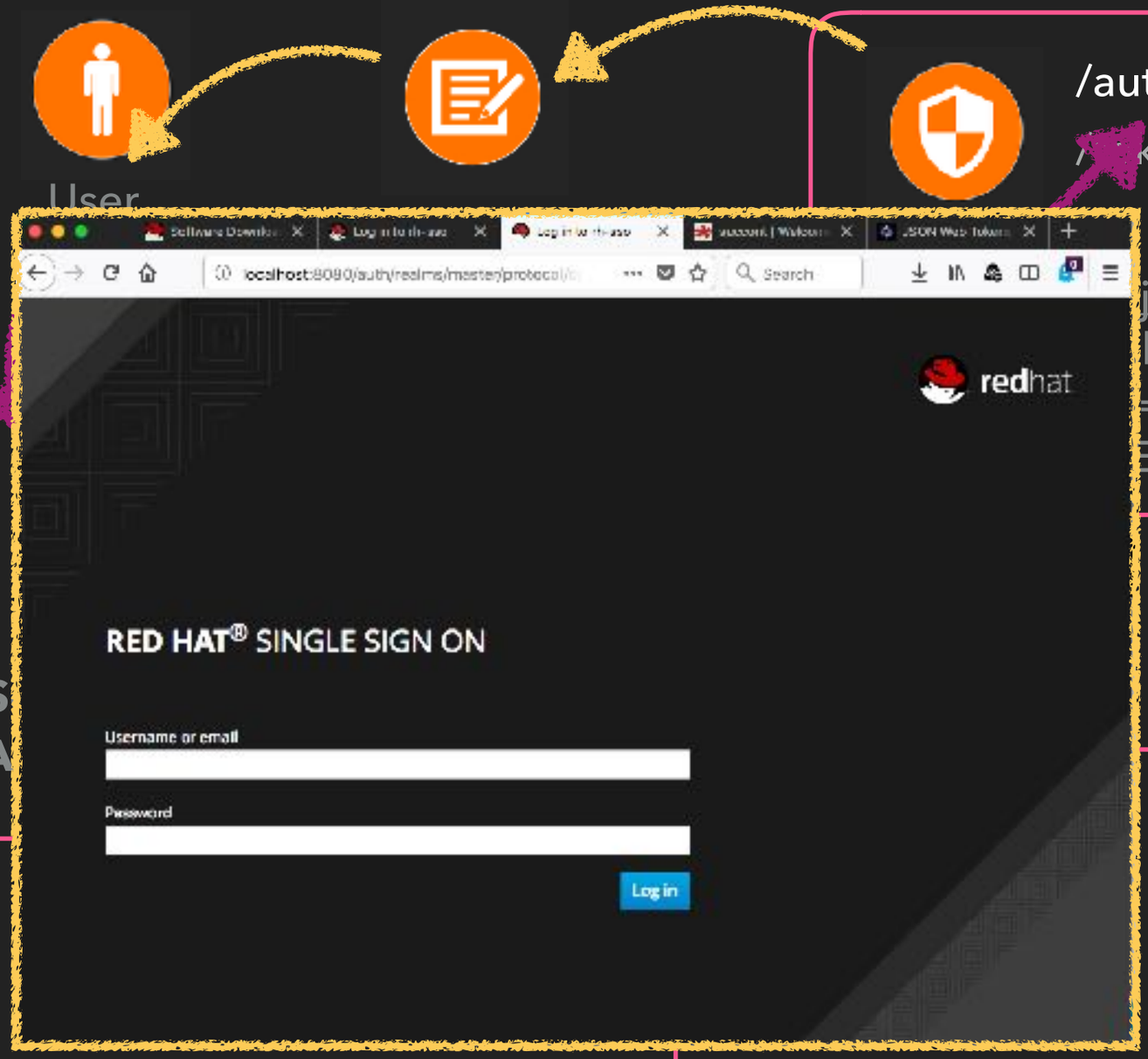
Service

client_id=askdjfkjk
secret=92kdksk3kd3k3d9w90
Auth Redirect = https://server.app.com/callback
Authorisation Endpoint = https://redhat-sso.com

#4 – AUTHORITY SERVER PROVIDES LOGIN FORM.



#4 – AUTHORITY SERVER PROVIDES LOGIN FORM.

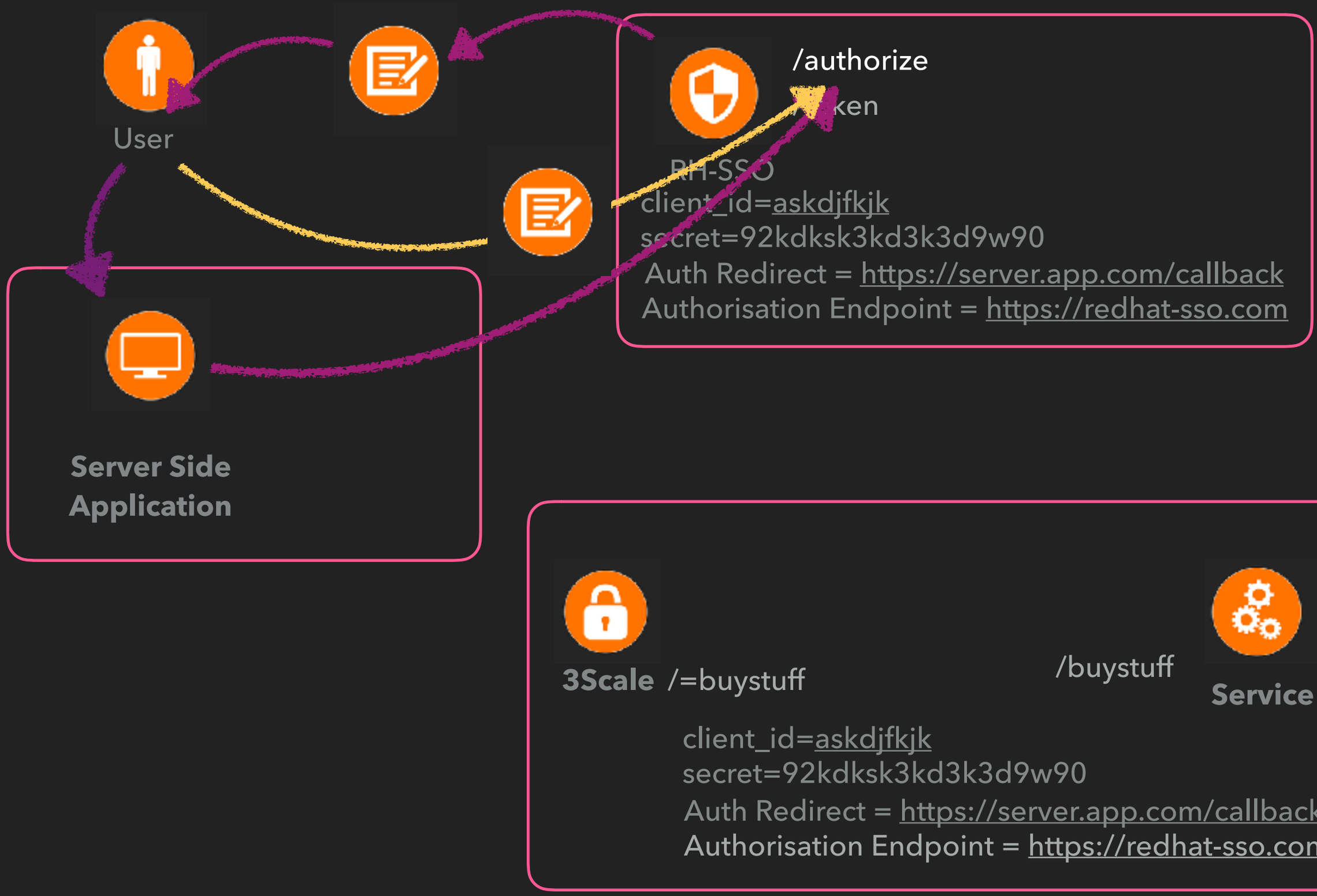


/buystuff

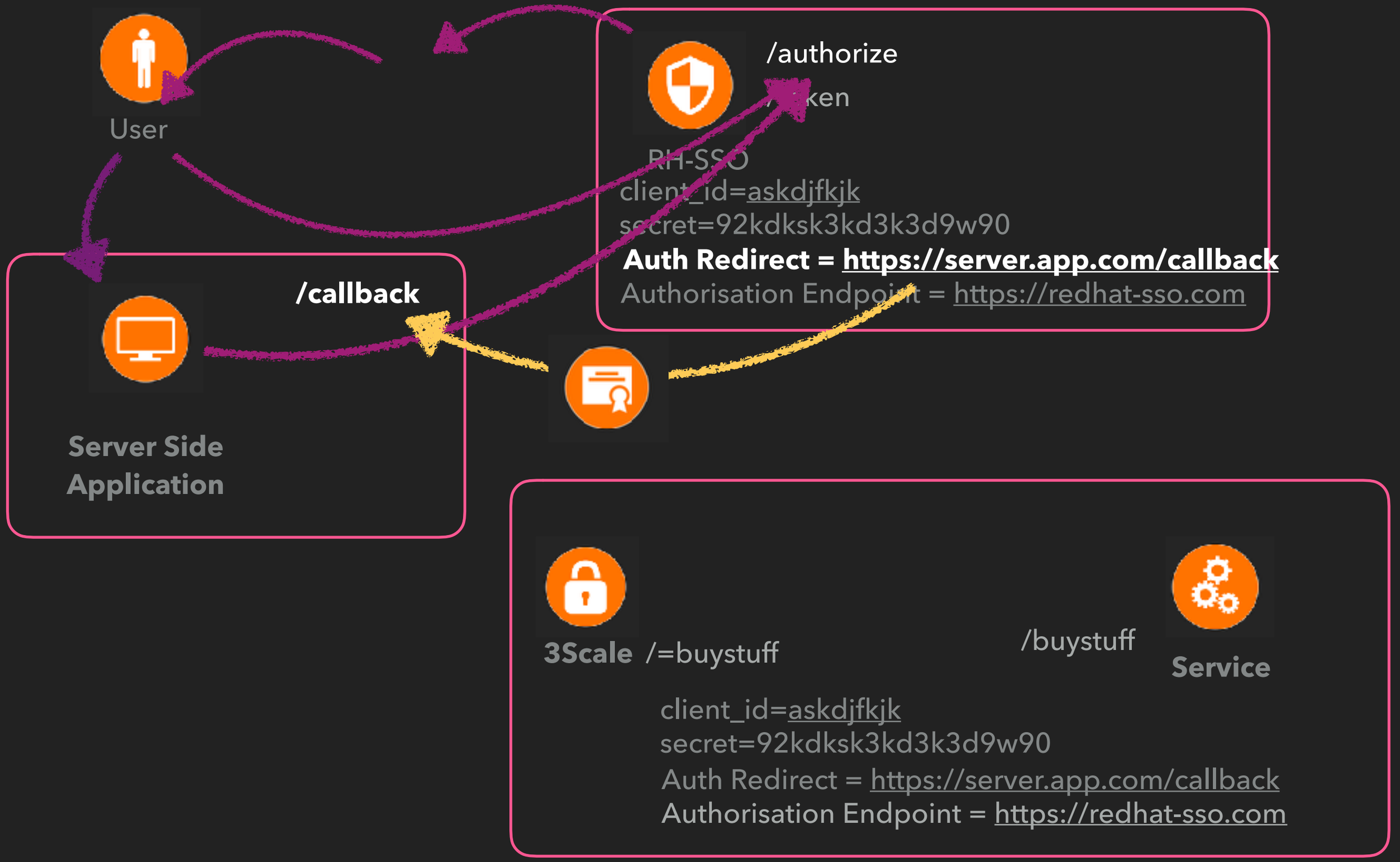
Service

client_id=askdjfkjk
secret=92kdksk3kd3k3d9w90
Auth Redirect = https://server.app.com/callback
Authorisation Endpoint = https://redhat-sso.com

#5 – THE USER NOW LOGINS IN TO THE AUTHORITY SERVER.



#6 – AUTHORITY SERVER NOW SENDS A TEMPORARY TOKEN.

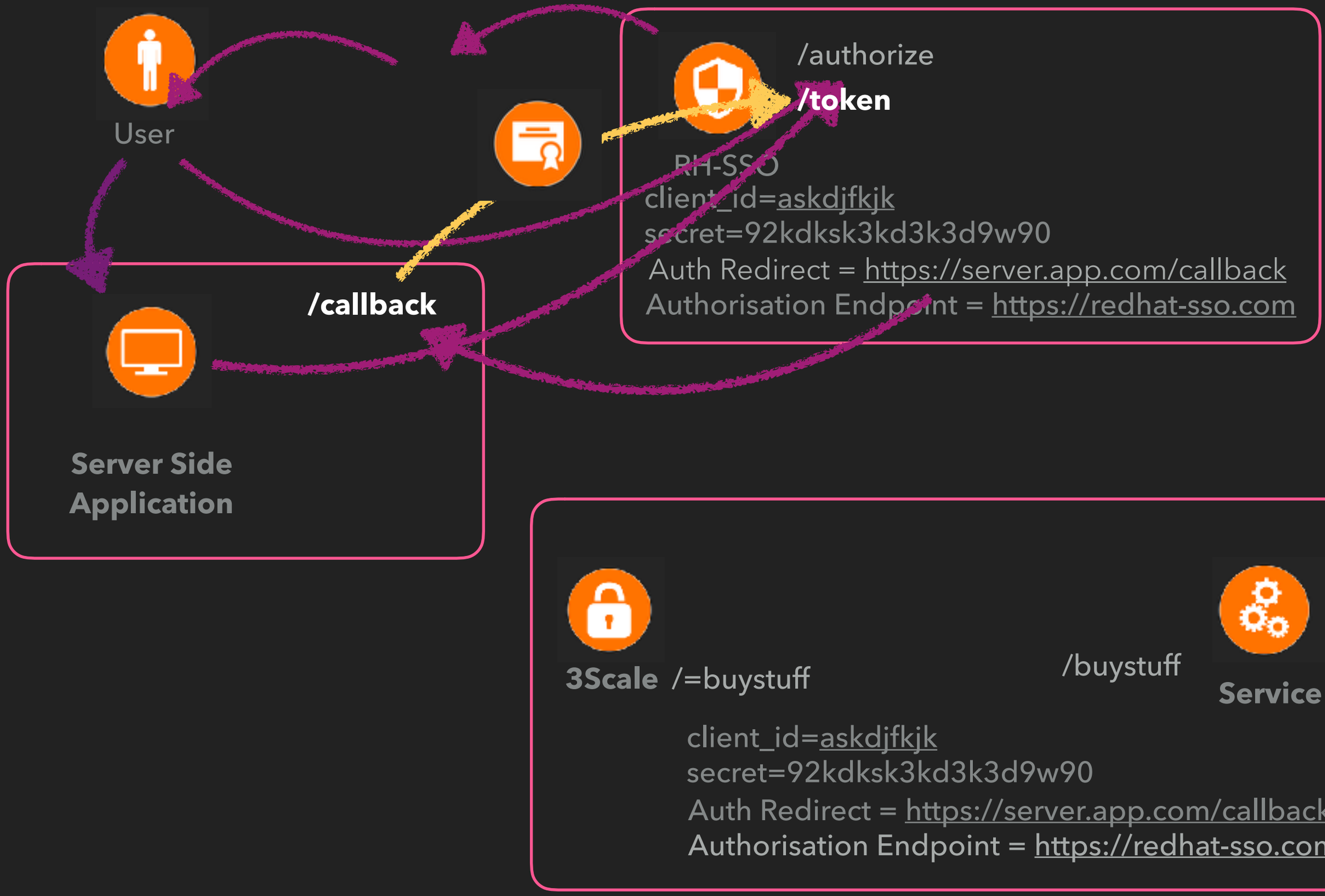


#6.1 – TEMPORARY AUTHORISATION TOKEN...

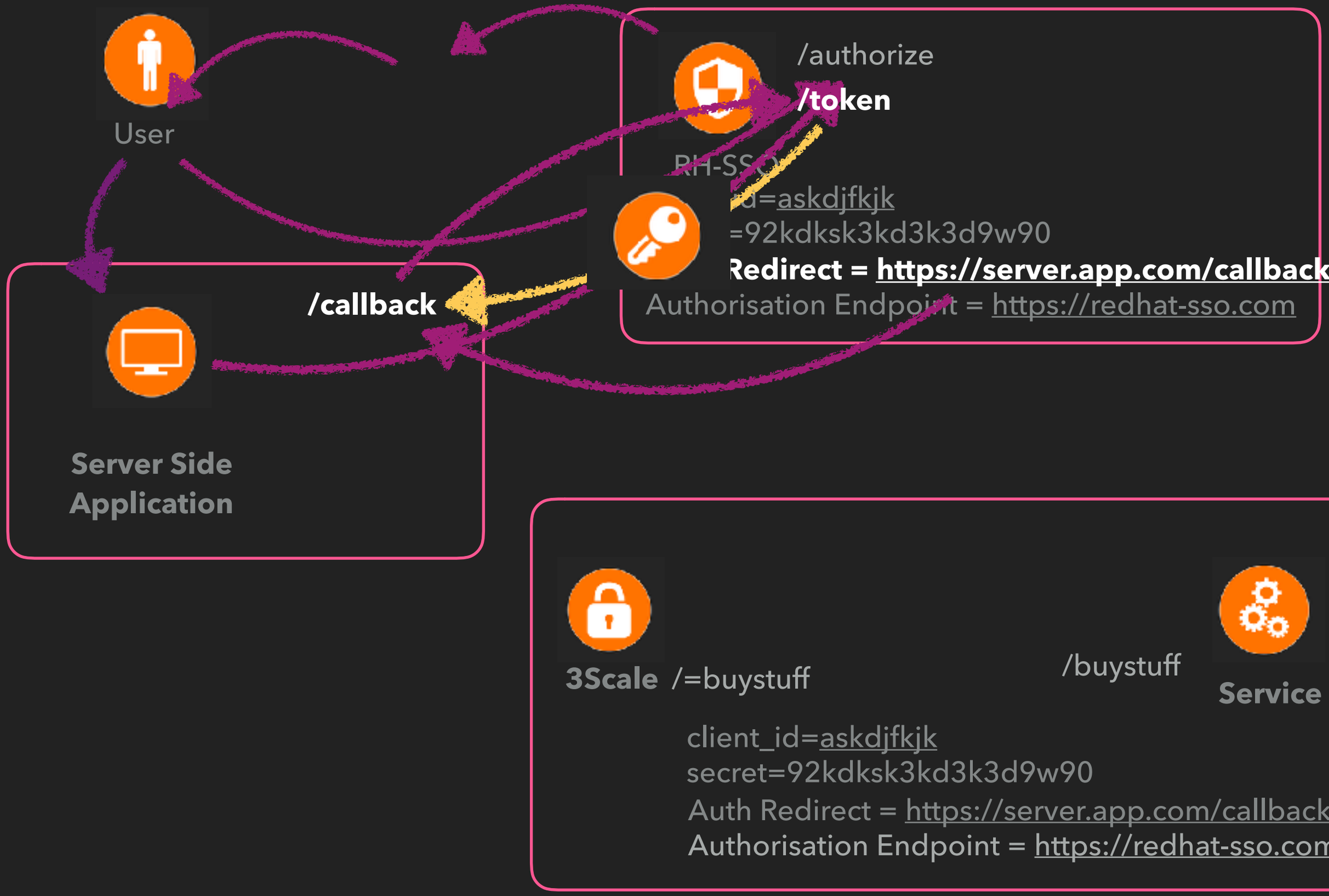
Is used to acquire an access code.

Think of this as being a nightclub cloakroom ticket - this can be used once only to acquire a **bearer** token.

#7 – CLIENT NOW CALLS THE GATEWAY'S TOKEN ENDPOINT..



#9 – AUTH. SERVER SENDS A VALID BEARER TOKEN.



#9.1 – DIVERSION ABOUT BEARER TOKENS…

- like money anyone who presents this - can spend it.

‘will pay to the bearer on demand’

#9.2 – DIVERSION ABOUT BEARER TOKENS...

- so what does a bearer token look like?

- **Authorization: Bearer**
  eyJhbGciOiJSUzI1NiJ9.eyJqdGkiOiJiY2IxMWY0OS1lNmFlLTQ0Y2EtYjA3Ny0zNzkyNTk0ZjBkOTgiLCJleHAiOjE0OTUyNzI3MzksIm5iZiI6MCwiaWF0IjoxNDk0MzIyMzM5LCJpc3MiOiJodHRwOi8vMDk2NmVhMWYubmdyb2suaW8vYXV0aC9yZWFsbXMvZm91cm1hcmtzIiwiYXVkIjoiNGQ2NTI0MDYiLCJzdWIiOiJkMjBkYzQxNS03NTJmLTRhNzktYTNhOC01MmU5NWVhNmRlYzYiLCJ0eXAiOiJCZWFyZXIiLCJhenAiOiI0ZDY1MjQwNiIsInNlc3Npb25fc3RhdGUiOiI1NWE4NDMyOS1jZjZkLTRiOWItYmE4Zi1hYmEwMzc2NGMyMWMiLCJjbGllbnRfc2Vzc2lvbiI6ImJmMWEwNzM5LWEzOWMtNDUxNS05YzAwLTc4ZTE4MjZiOGQzNiIsImFsbG93ZWQtb3JpZ2lucyI6WyJodHRwczovL3d3dy5nZXRwb3N0bWFuLmNvbSJdLCJyZWFsbV9hY2Nlc3MiOnsicm9sZXMiOlsiYWNjZXNzX215X3Jlc291cmNlIl19LCJyZXNvdXJjZV9hY2Nlc3MiOnsiYWNjb3VudCI6eyJyb2xlcyI6WyJtYW5hZ2UtYWNjb3VudCIsInZpZXctcHJvZmlsZSJdfX0sInNjb3BlIjoib3BlbmlkIGVtYWlsIHByb2ZpbGUiLCJlbWFpbF92ZXJpZmllZCI6dHJ1ZSwibmFtZSI6IkpvaG4gRG9lIn0.Ebvi9PlEpHX7Jczz_FTottHiTvz2n82vnp-1LbxeQMEwAPJTaipJ77OCVuKhdlQNb4YmzzYp3cKXh2MOKid7bgZRzYb4-ZWk5RoTq43gEei_U1tz_PNzRDJi7O_emZRwRSPHfwgWflQkHJ-jhqDBv9_HNjIXSnqdN71l5KxZB7jCgVur-aBntph3XbQpM76zSgO4qkln0MFkk2xh8Y9353hBLi5AqpxahmZ5Ba_k3LbXeg8WS_2G1QsexL4SzcJneo1W4ZmfrKoMpmBJi_QXJIEO7QKjkwgUsSowctCOpe9XnCgHJVL6V24i7cvL3mB6oDnL0mquuZ9404D1Eo7mbQ

- Accept: */*

- Postman-Token: 86b86d4a-8369-40af-8612-9f0d3589fdfb

- Cf-Ray: 35c3a94bb1ac35ae-LHR

- X-3Scale-Proxy-Secret-Token: Shared_secret_sent_from_proxy_to_API_backend_169ad455fe40801e

-

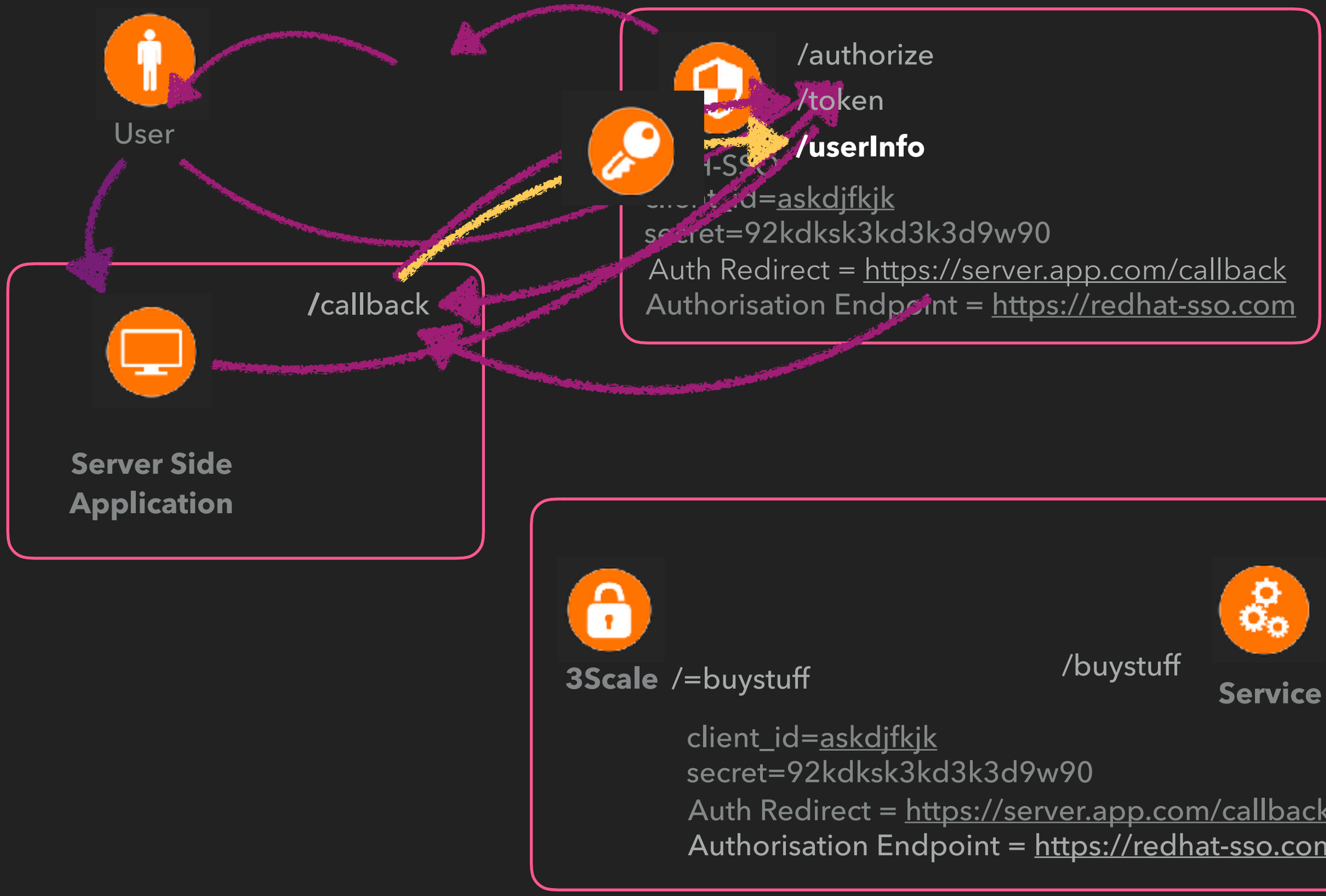
#9.3 – DIVERSION ABOUT BEARER TOKENS…

- if you base64 decrypt you get:

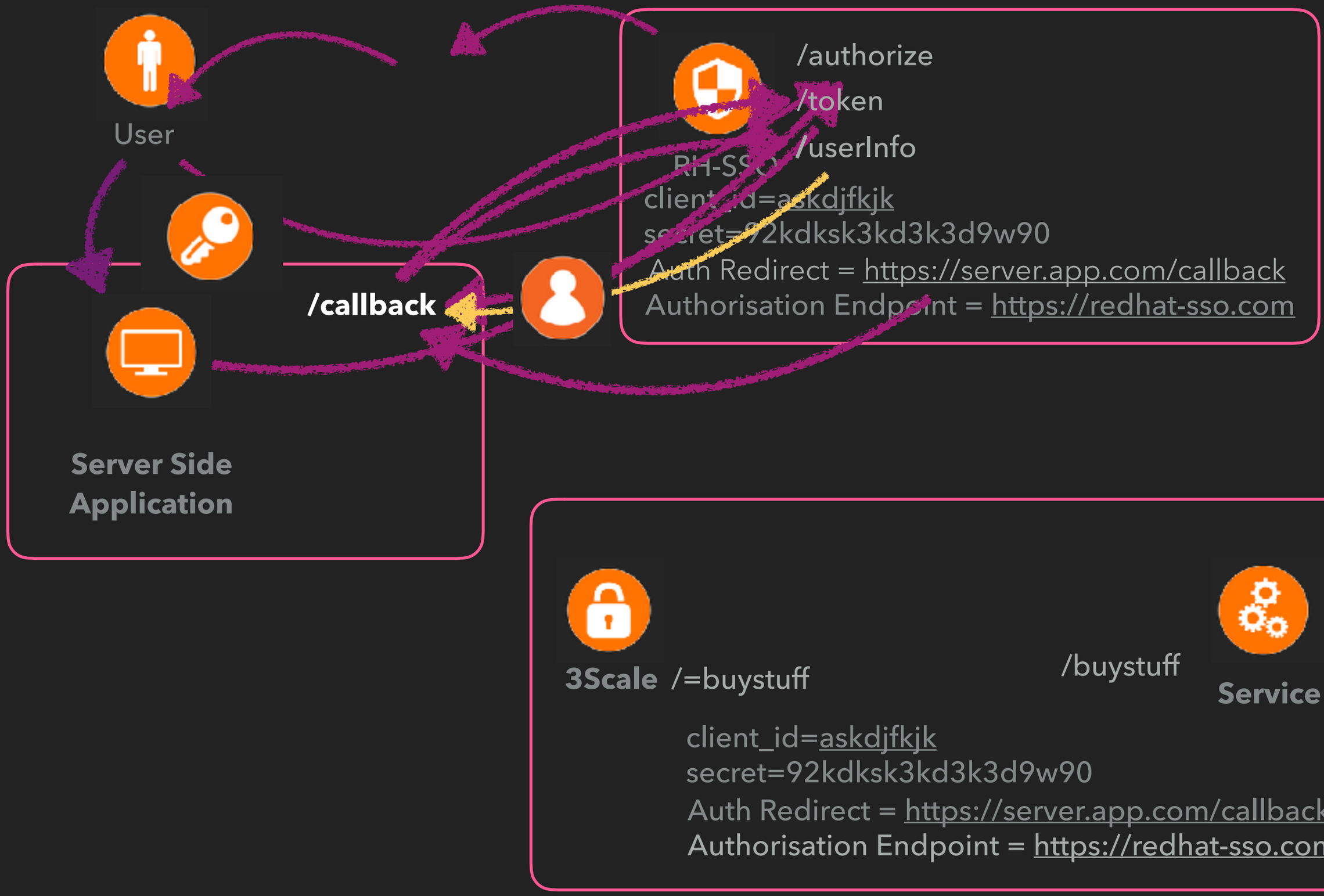
```
AUTHORIZATION: BEARER
{
 "JTI": "BCB11F49-E6AE-44CA-B077-3792594F0D98",
 "EXP": 1495272739,
 "NBF": 0,
 "IAT": 1494322339,
 "ISS": "HTTP://0966EA1F.NGROK.IO/AUTH/REALMS/FOURMARKS",
 "AUD": "4D652406",
 "SUB": "D20DC415-752F-4A79-A3A8-52E95EA6DEC6",
 "TYP": "BEARER",
 "AZP": "4D652406",
 "SESSION_STATE": "55A84329-CF6D-4B9B-BA8F-ABA03764C21C",
 "CLIENT_SESSION": "BF1A0739-A39C-4515-9C00-78E1826B8D36",
 "ALLOWED-ORIGINS": [
  "HTTPS://WWW.GETPOSTMAN.COM"
 ],
 "REALM_ACCESS": {
  "ROLES": [
   "ACCESS_MY_RESOURCE"
  ]
 },
 "RESOURCE_ACCESS": {
  "ACCOUNT": {
   "ROLES": [
    "MANAGE-ACCOUNT",
    "VIEW-PROFILE"
   ]
  }
 },
 "NAME": "TEST USER",
 "PREFERRED_USERNAME": "TESTUSER",
 "GIVEN_NAME": "TEST",
 "FAMILY_NAME": "USER",
 "EMAIL": "TEST@BLAH.COM"
}
```

- **notice the role information**
- **the token is a ‘JWT’ - pronounce it ‘JOT’.**

#9 – SUBMIT BEARER TOKEN TO GET AN ID TOKEN.



#10 – IDENTITY TOKEN GETS RETURNED . . .



#11 - ID_TOKEN...

Digitally signed by the Auth Server.

A Standardised Identity token.

Obtained by calling the

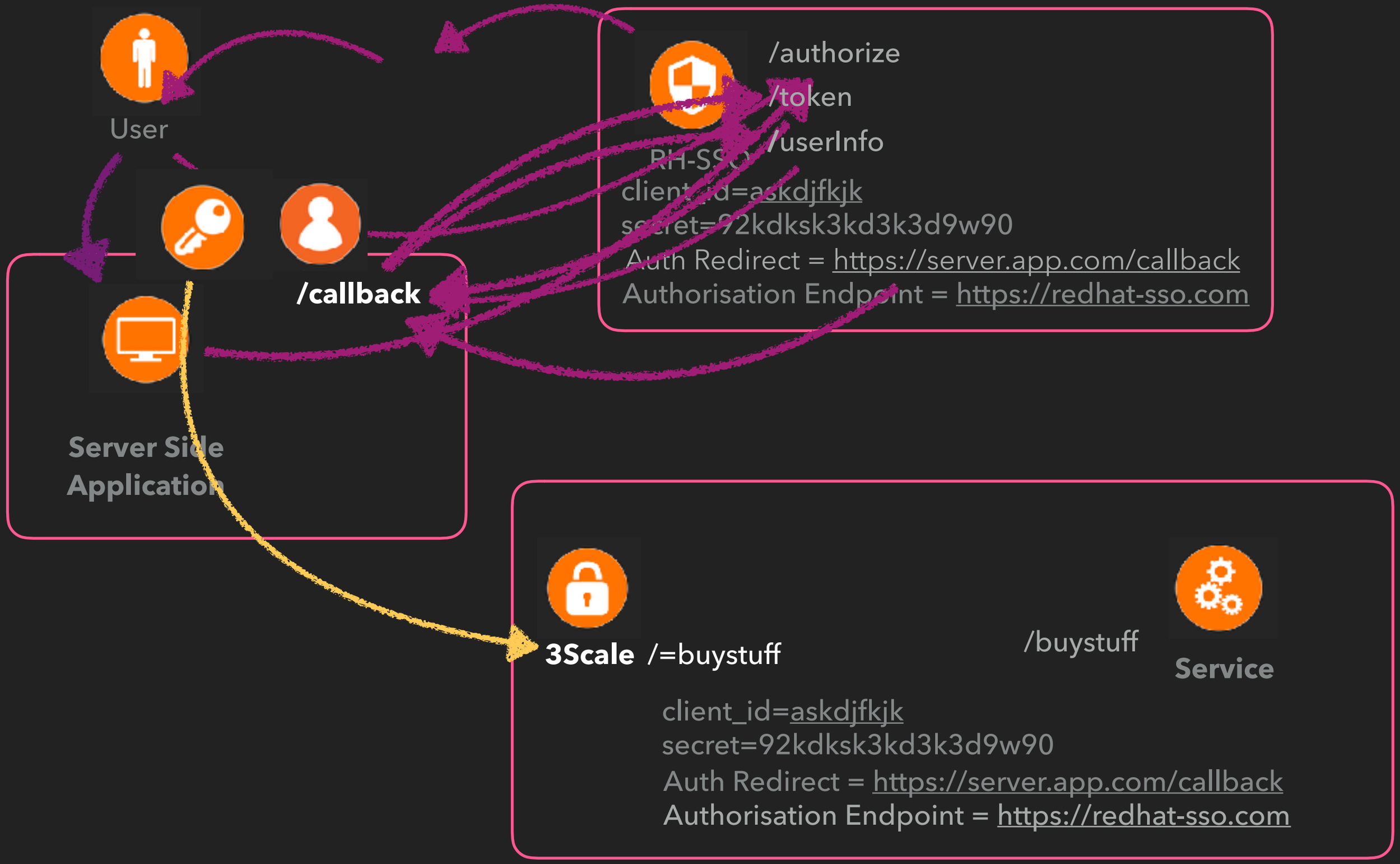
userInfo endpoint.

#11 - ID_TOKEN...

@todo Show identity

token - decrypted.

#10 –BEARER TOKEN GETS SUBMITTED...



3SCALE GATEWAY VALIDATES THE BEARER TOKEN...

- Checks the timestamp for ‘expired’ token.
- Checks the client_id is still active.
- Checks that the token came from the expected domain
- Validates the token using the public key of the Identity server.

#9.3 – DIVERSION ABOUT BEARER TOKENS...

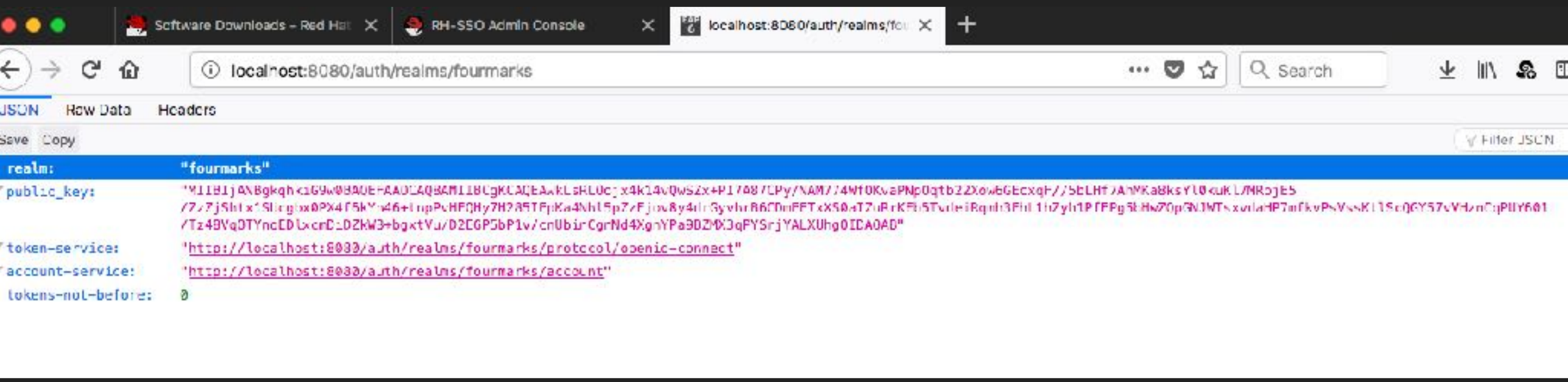
```
AUTHORIZATION: BEARER
{
 "JTI": "BCB11F49-E6AE-44CA-B077-3792594F0D98",
 "EXP": 1495272739,
 "NBF": 0,
 "IAT": 1494322339,
 "ISS": "HTTP://0966EA1F.NGROK.IO/AUTH/REALMS/FOURMARKS",
 "AUD": "4D652406",
 "SUB": "D20DC415-752F-4A79-A3A8-52E95EA6DEC6",
 "TYP": "BEARER",
 "AZP": "4D652406",
 "SESSION_STATE": "55A84329-CF6D-4B9B-BA8F-ABA03764C21C",
 "CLIENT_SESSION": "BF1A0739-A39C-4515-9C00-78E1826B8D36",
 "ALLOWED-ORIGINS": [
  "HTTPS://WWW.GETPOSTMAN.COM"
 ],
 "REALM_ACCESS": {
  "ROLES": [
   "ACCESS_MY_RESOURCE"
  ]
 },
 "RESOURCE_ACCESS": {
  "ACCOUNT": {
   "ROLES": [
    "MANAGE-ACCOUNT",
    "VIEW-PROFILE"
   ]
  }
 },
 "NAME": "TEST USER",
 "PREFERRED_USERNAME": "TESTUSER",
 "GIVEN_NAME": "TEST",
 "FAMILY_NAME": "USER",
 "EMAIL": "TEST@BLAH.COM"
}
```

- **The identity server / realm is hyperlinked to the SSO server's public key.**

#9.3 – DIVERSION ABOUT BEARER TOKENS...

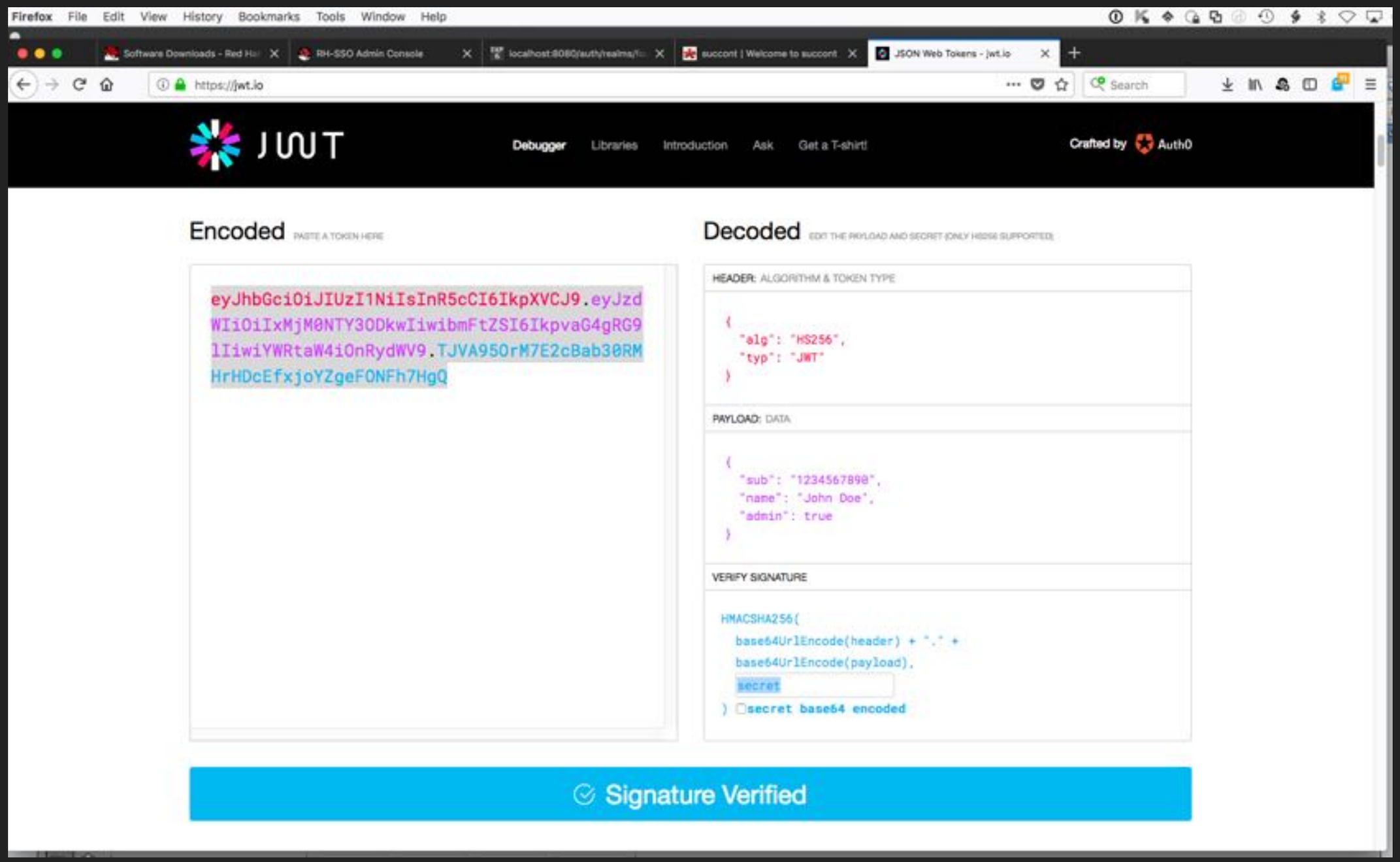
- The public key of the identity server.

>> http://0966ea1f.ngrok.io/auth/realms/fourmarks

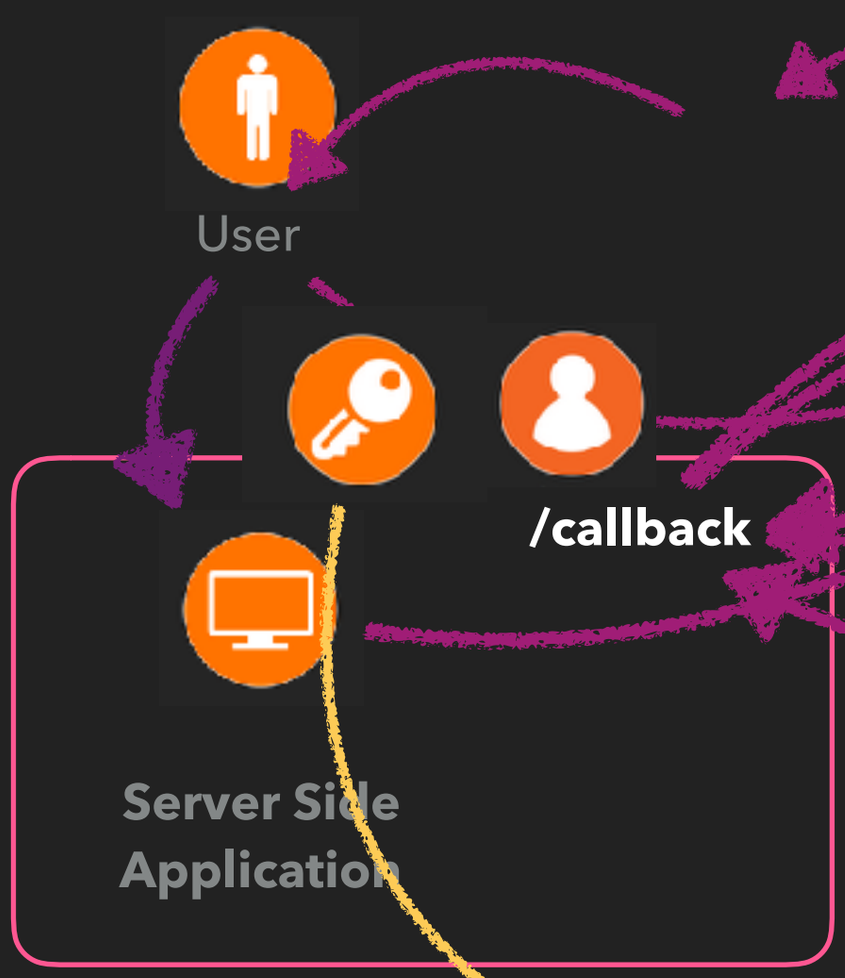


#9.3 – DIVERSION ABOUT BEARER TOKENS...

- Validating the JWT has been signed using the private key.



#10 – 3SCALE GATEWAY ACTS AS A FORTRESS...



3Scale /=buystuff

/buystuff

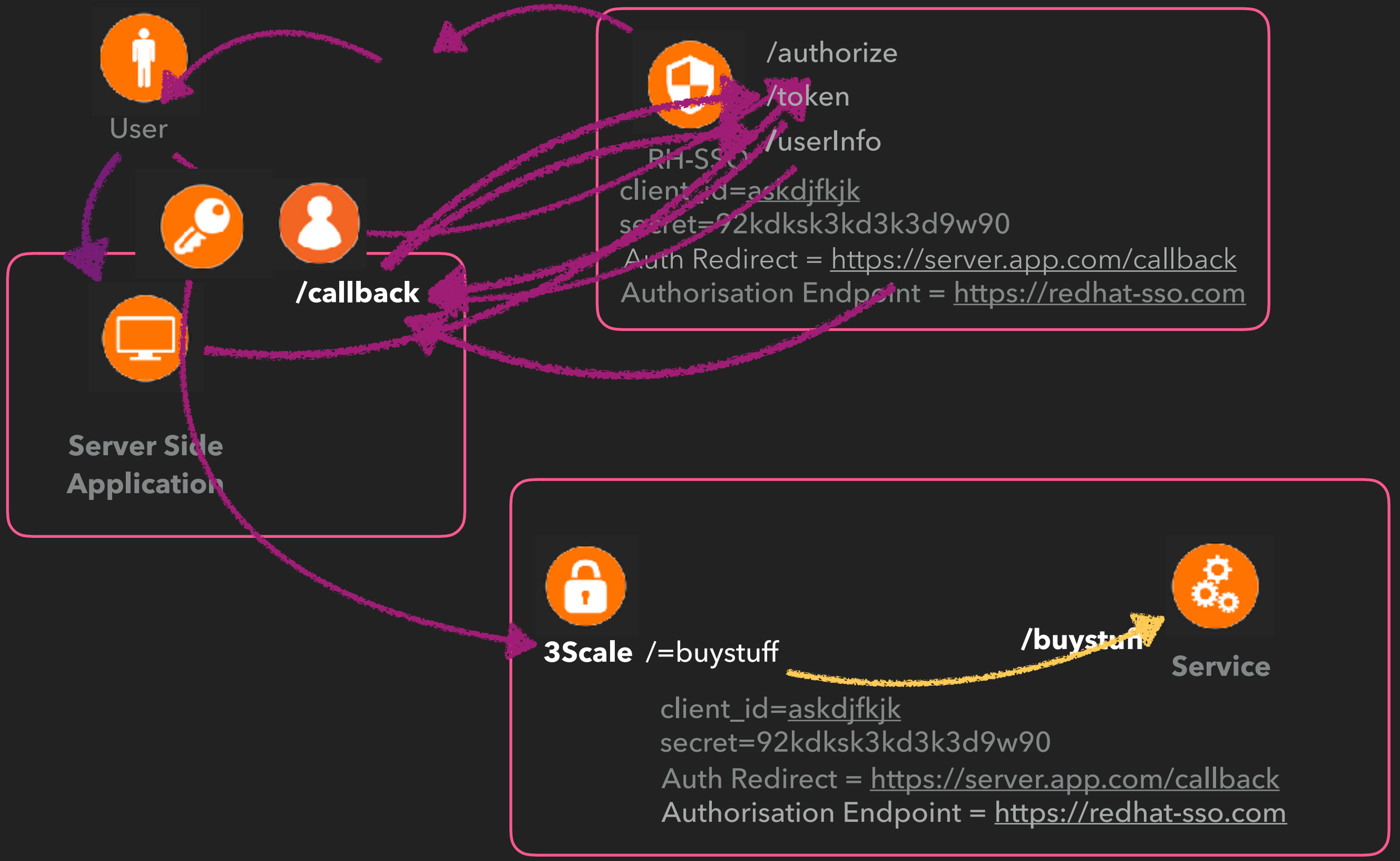
Service

client_id=askdjfkjk
secret=92kdksk3kd3k3d9w90
Auth Redirect = https://server.app.com/callback
Authorisation Endpoint = https://redhat-sso.com

#9.3 – ARCHITECTURAL BENEFITS OF 3SCALE...

- Security is applied in a consistent fashion.
- Services behind the proxy do not need to worry about security.
- Services remain easier to debug..
- No tight coupling between clients and services (reverse proxy pattern).

#10 –BEARER TOKEN FINALLY GETS AND WE CAN ‘BUY STUFF’ . . .




3scale
by Red Hat®