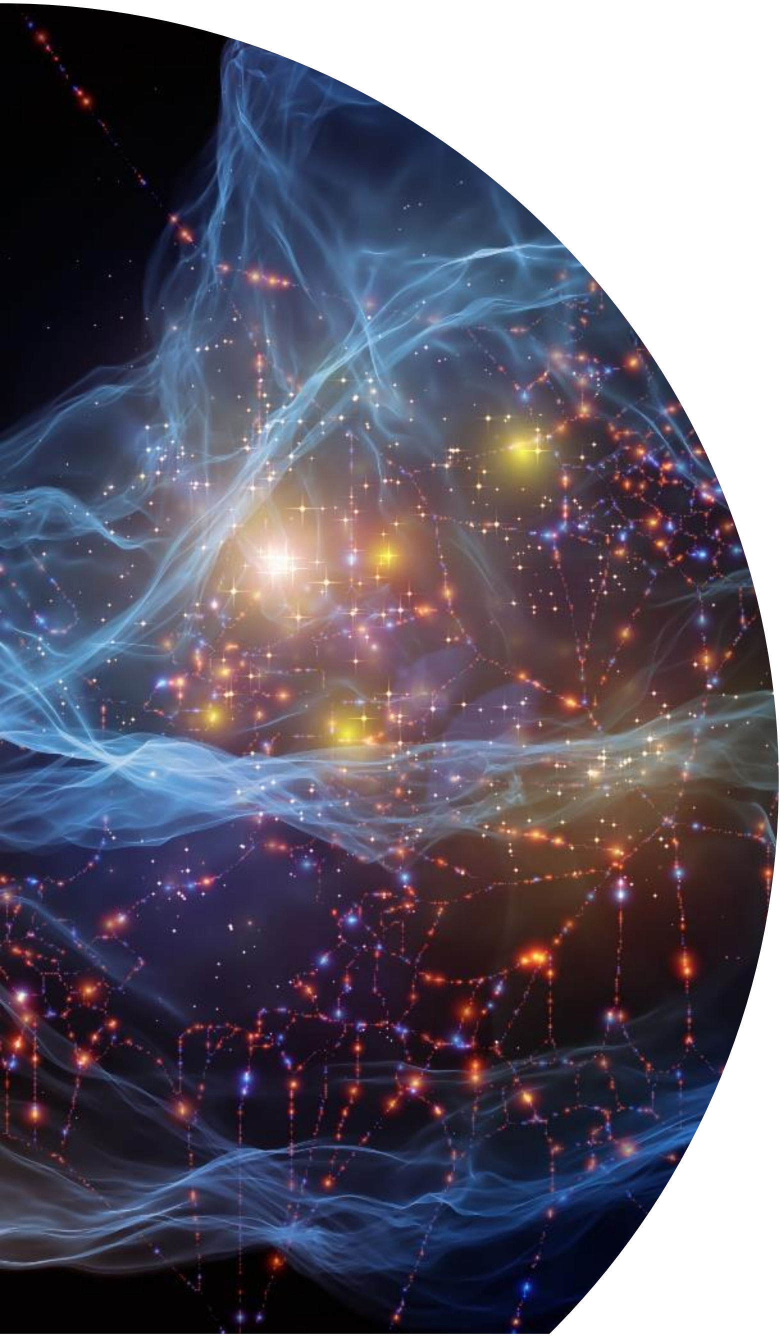
OpenCAPIF

by ETSI

OCF Hackfest

Jorge Moratinos, Pelayo Torres and Stavros Charismiadis

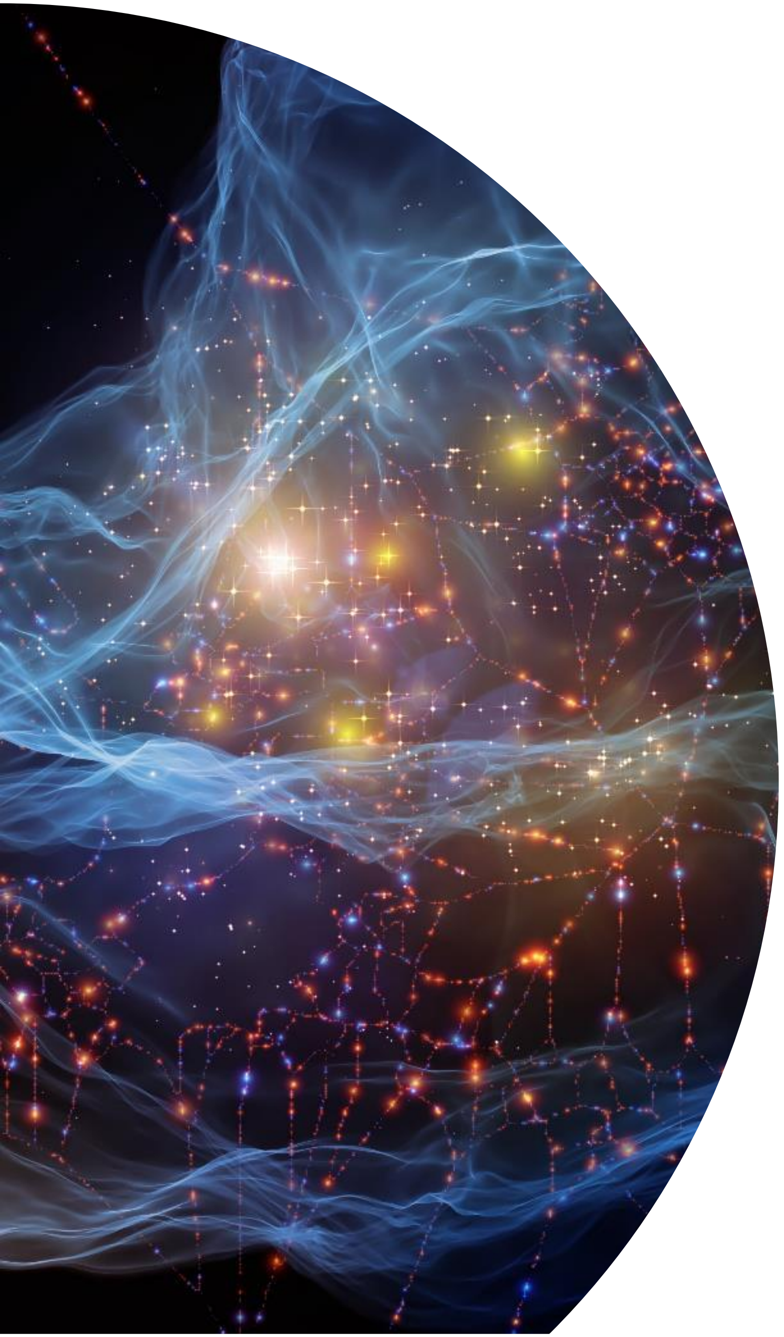
14/11/25



Agenda

OpenCAPIF

- Welcome & Logistics
- Preparation of attendees.
- Run Locally OpenCAPIF.
- Coffee Break & Group picture



Agenda

OpenCAPIF

- Verify if OpenCAPIF is working.
- User Registration Flow by Administrator.
- Provider Onboarding flow by customer User.
- Invoker Onboarding flow by customer User.

Today's Presenters

Stavros Charismiadis

OCF TSC Member

Pelayo Torres

OCF TSC Member

Jorge Moratinos

OCF TSC Chair

Preparation of Attendees

Preparation of Attendees

All information and URLs used are present on https://labs.etsi.org/rep/groups/ocf/-/wikis/OCF-HACKFEST-1 agenda.

In order to simplify we prepare 2 Ubuntu VMs, one for amd/intel processors and other one for Mac M1/M2/M3 processors.

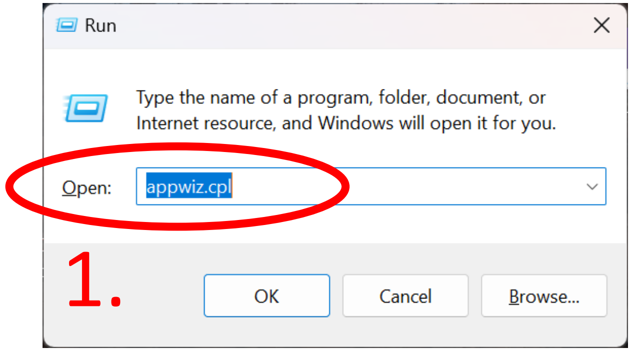
- If your laptop is amd/intel processor you will need to download VirtualBox application and amd64 VM:
  - https://www.virtualbox.org/wiki/Downloads
  - VirtualBox Hackfest Ubuntu AMD64 VM (https://drive.google.com/file/d/1DyYURh6a3qrffvENxj7h1cTG3vqs5Z67/view?usp=sharing)
- If your laptop is Mac with M1/M2/M3 processor, you will need to download UTM and ARM VM:
  - https://mac.getutm.app/
  - UTM Hackfest Ubuntu ARM64 VM (https://drive.google.com/file/d/1ThDFYljbeYxJKKmvKHqCWwegCYKHH2NA/view?usp=sharing)

NOTE: Confirm that CPU supports AVX (Advanced Vector Extensions). On Windows execute the following command to enable: *“bcdedit /set xsavedisable 0”*

Preparation of Attendees

If your laptop runs Windows 11 (or 10) you will need to turn off virtualization (Hyper-V) features:

- Press Win+R. In the Run field, type appwiz.cpl and press Enter.
- Turn Windows features on or off.
- Deselect the following checkboxes (or Hyper-V checkbox for oldest versions)
- Restart laptop

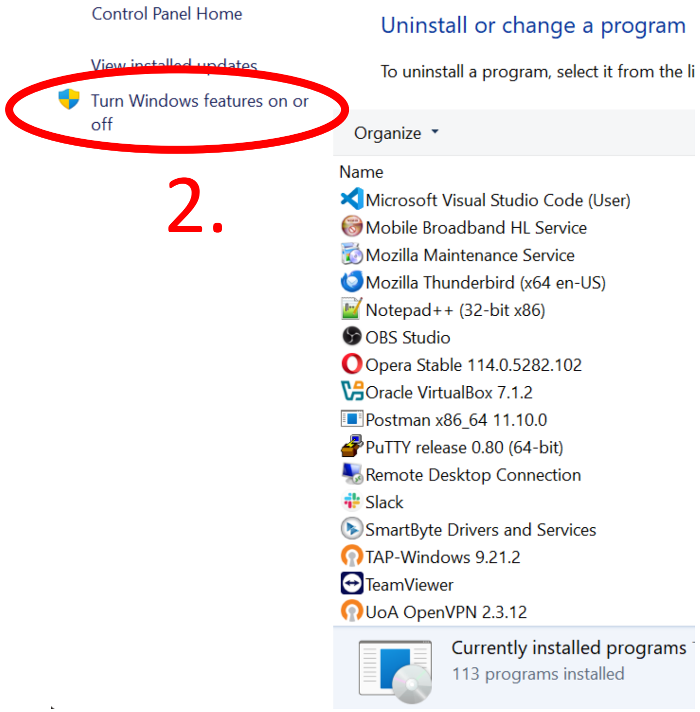


1.

Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: appwiz.cpl



2.

Control Panel Home

View installed updates

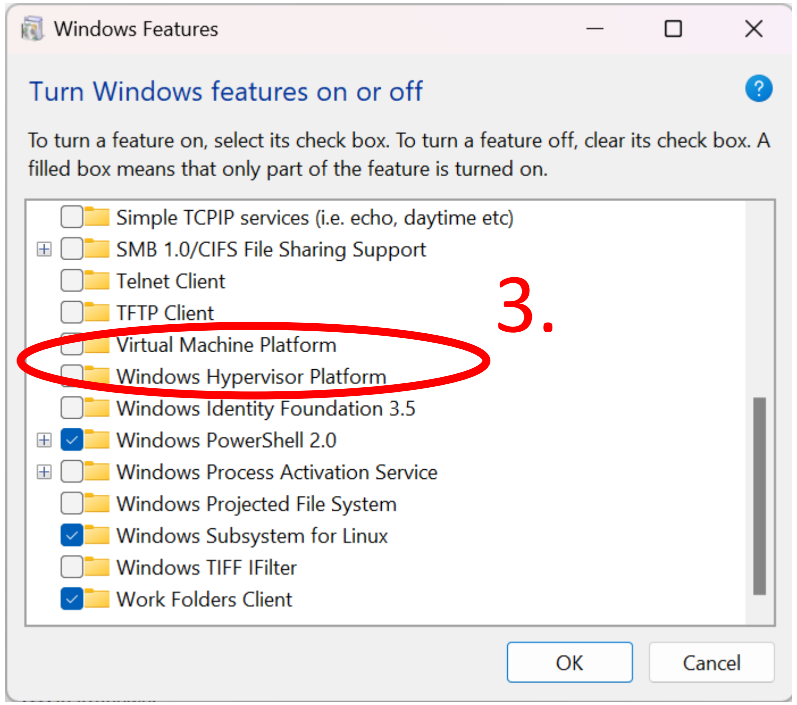
Turn Windows features on or off

Uninstall or change a program

To uninstall a program, select it from the li

Microsoft Visual Studio Code (User)
Mobile Broadband HL Service
Mozilla Maintenance Service
Mozilla Thunderbird (x64 en-US)
Notepad++ (32-bit x86)
OBS Studio
Opera Stable 114.0.5282.102
Oracle VirtualBox 7.1.2
Postman x86_64 11.10.0
PuTTY release 0.80 (64-bit)
Remote Desktop Connection
Slack
SmartByte Drivers and Services
TAP-Windows 9.21.2
TeamViewer
UoA OpenVPN 2.3.12

Currently installed programs
113 programs installed



3.

Windows Features

Turn Windows features on or off

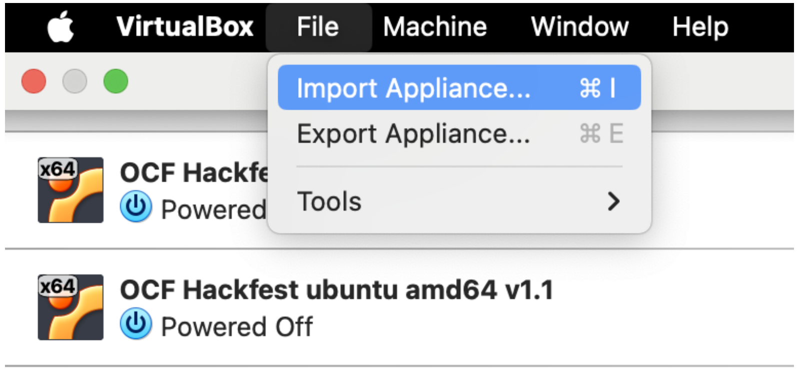
To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

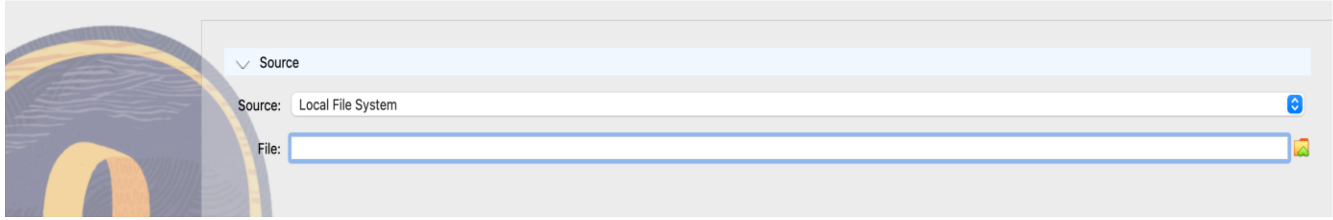
- ☐ Simple TCPIP services (i.e. echo, daytime etc)
- ☐ SMB 1.0/CIFS File Sharing Support
- ☐ Telnet Client
- ☐ TFTP Client
- ☐ Virtual Machine Platform
- ☐ Windows Hypervisor Platform
- ☐ Windows Identity Foundation 3.5
- ☑ Windows PowerShell 2.0
- ☐ Windows Process Activation Service
- ☐ Windows Projected File System
- ☑ Windows Subsystem for Linux
- ☐ Windows TIFF IFilter
- ☑ Work Folders Client

Preparation of Attendees

Import AMD64 VM on VirtualBox:

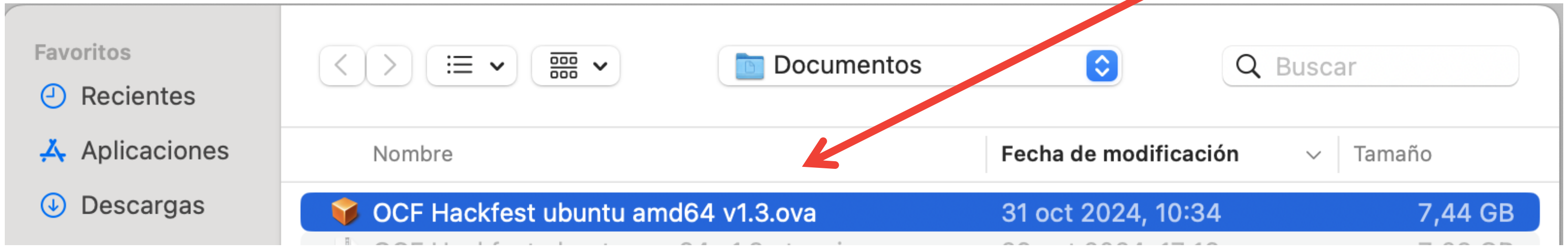
- File -> Import Appliance:





Source: Local File System

File:

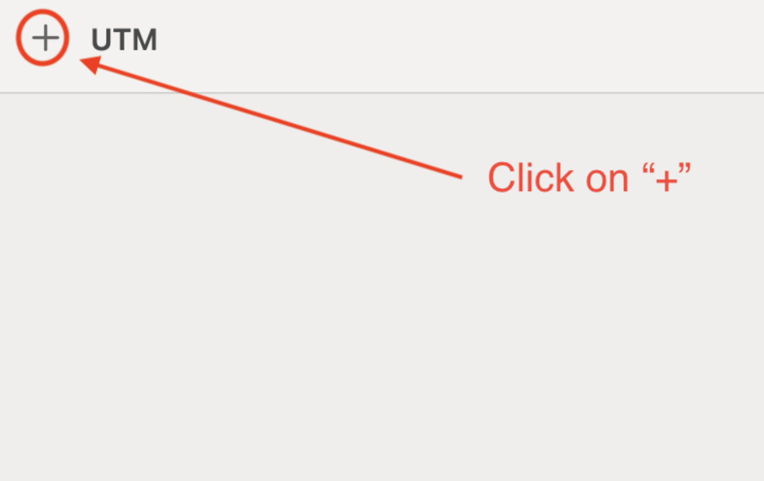


Nombre	Fecha de modificación	Tamaño
OCF Hackfest ubuntu amd64 v1.3.ova	31 oct 2024, 10:34	7,44 GB

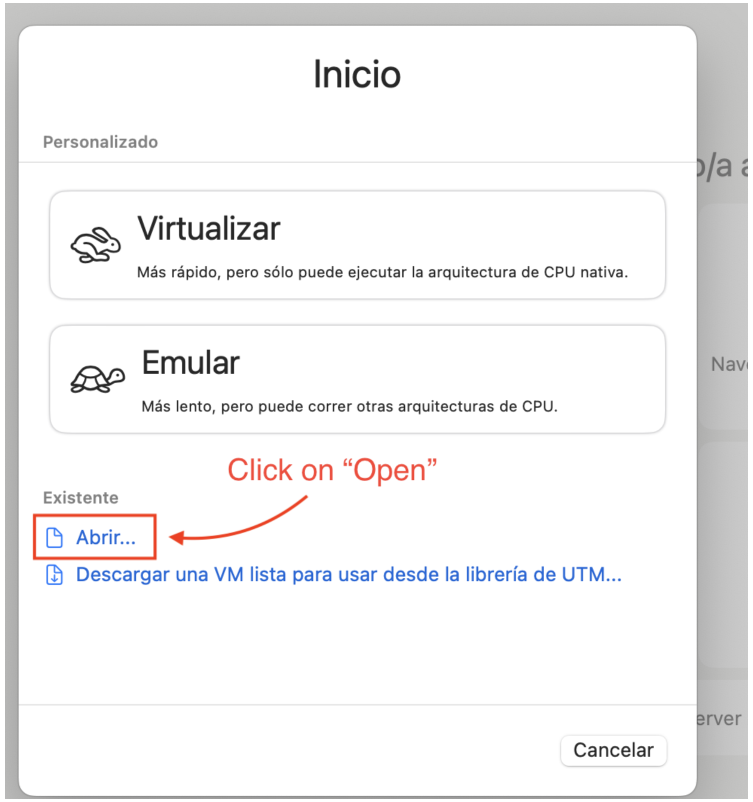
Preparation of Attendees

Import ARM64 VM on UTM:

- Click on “+” and open VM file:



Click on “+”



Inicio

Personalizado

Virtualizar

Más rápido, pero sólo puede ejecutar la arquitectura de CPU nativa.

Emular

Más lento, pero puede correr otras arquitecturas de CPU.

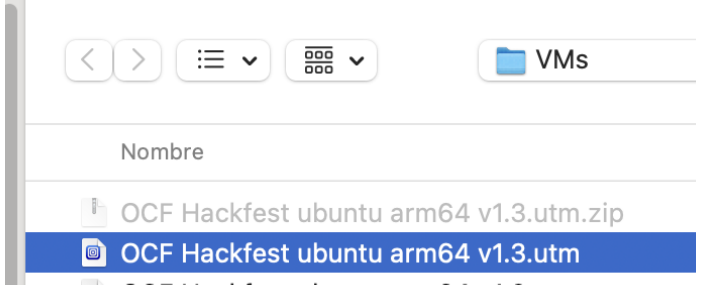
Existente

Abrir...

Descargar una VM lista para usar desde la librería de UTM...

Cancelar

Click on “Open”



Nombre

OCF Hackfest ubuntu arm64 v1.3.utm.zip

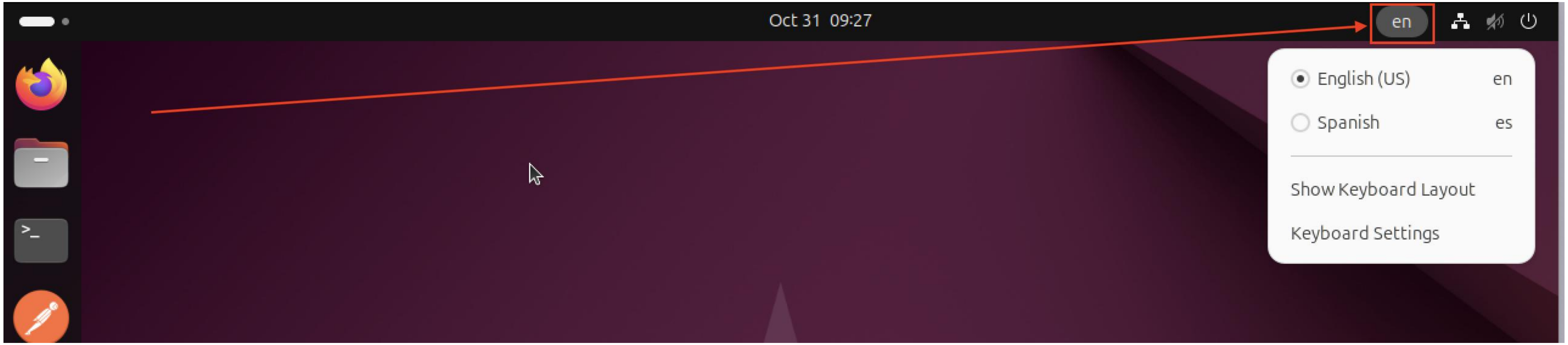
OCF Hackfest ubuntu arm64 v1.3.utm

Preparation of Attendees

Run VM and setup keyboard:

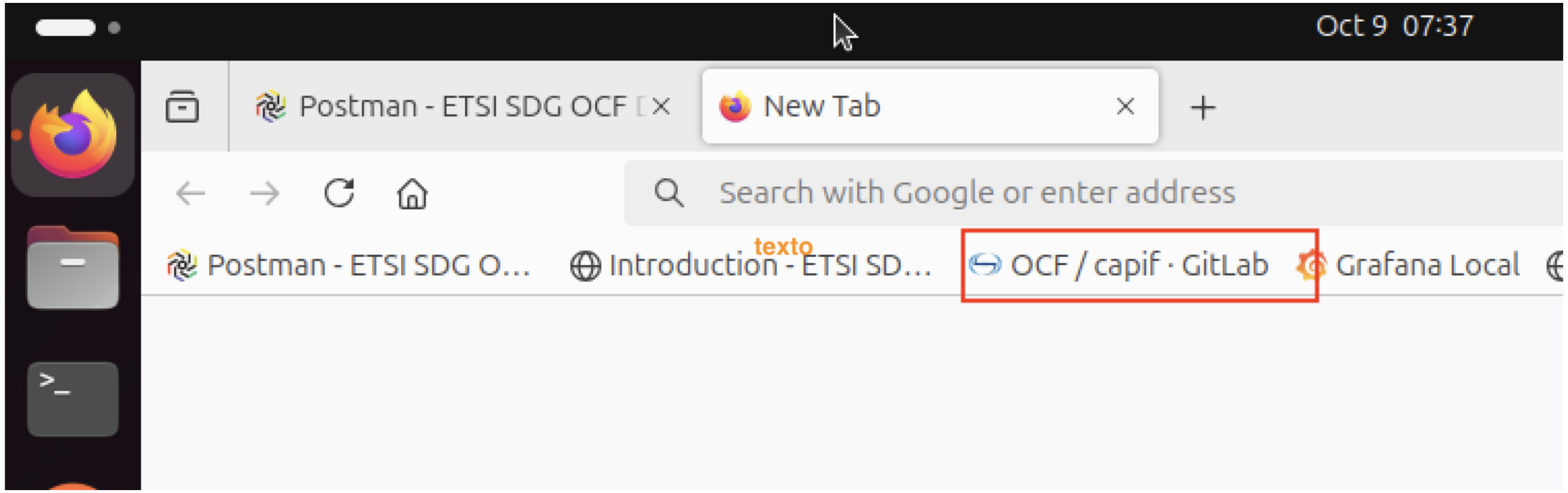
- Login in Ubuntu with next credentials:
  - User: ocf
  - Password: ocf

After login setup your keyboard according to your



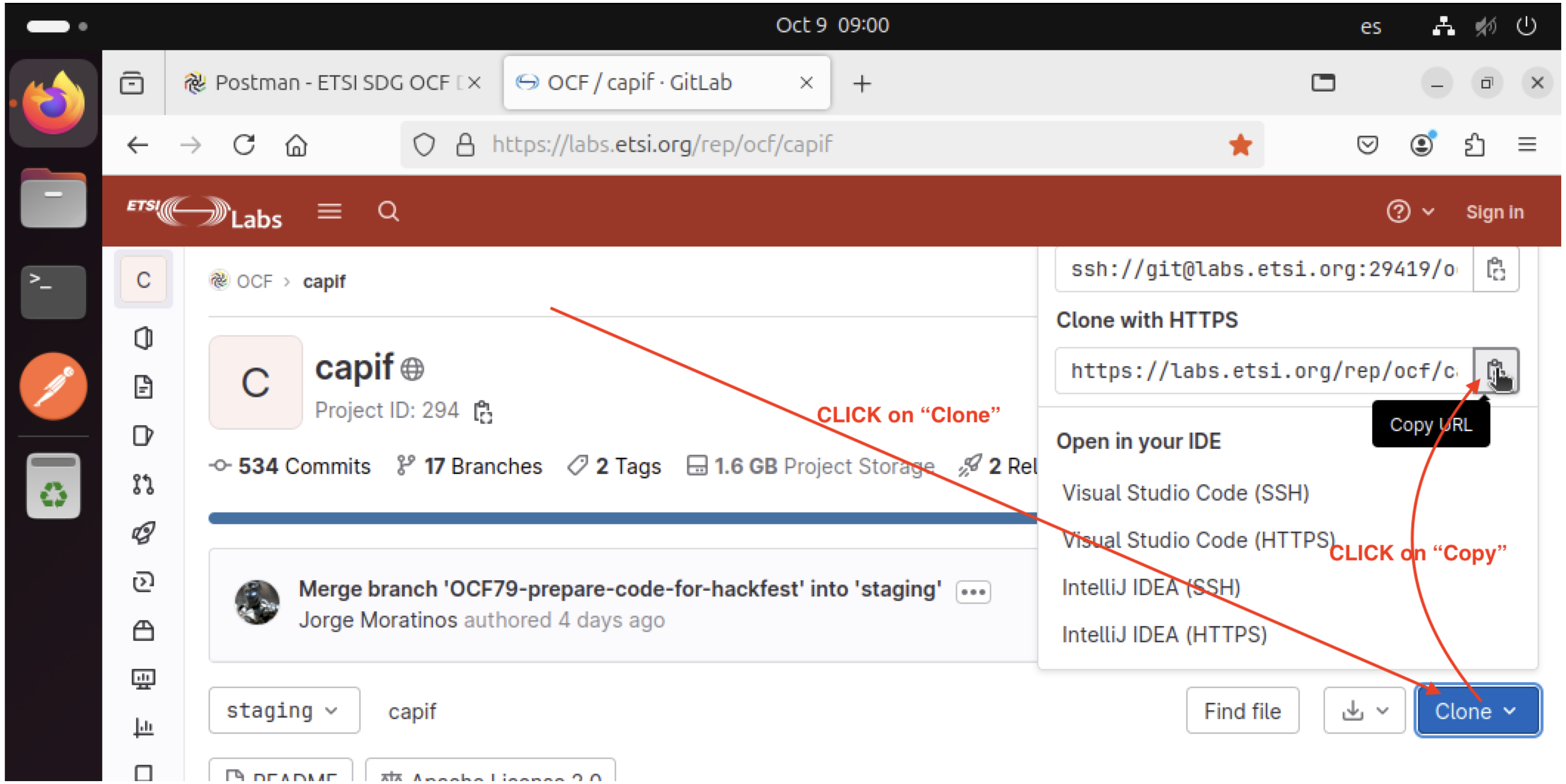
Preparation of Attendees

We can start opening ”Firefox”, click on “+” on tabs and go to “OCF/capif – GitLab”:



Preparation of Attendees

Copy url for clone:

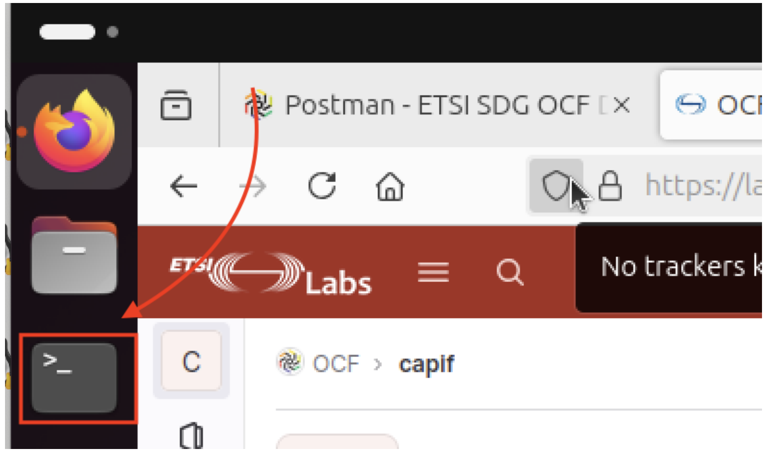


CLICK on “Clone”

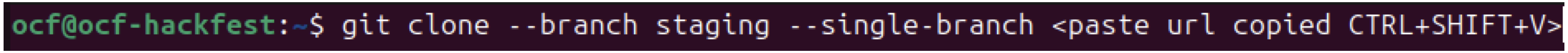
CLICK on “Copy”

Preparation of Attendees

Open Terminal on left:



Write next command to clone current staging repository:



```
ocf@ocf-hackfest:~$ git clone --branch staging --single-branch <paste url copied CTRL+SHIFT+V>
```

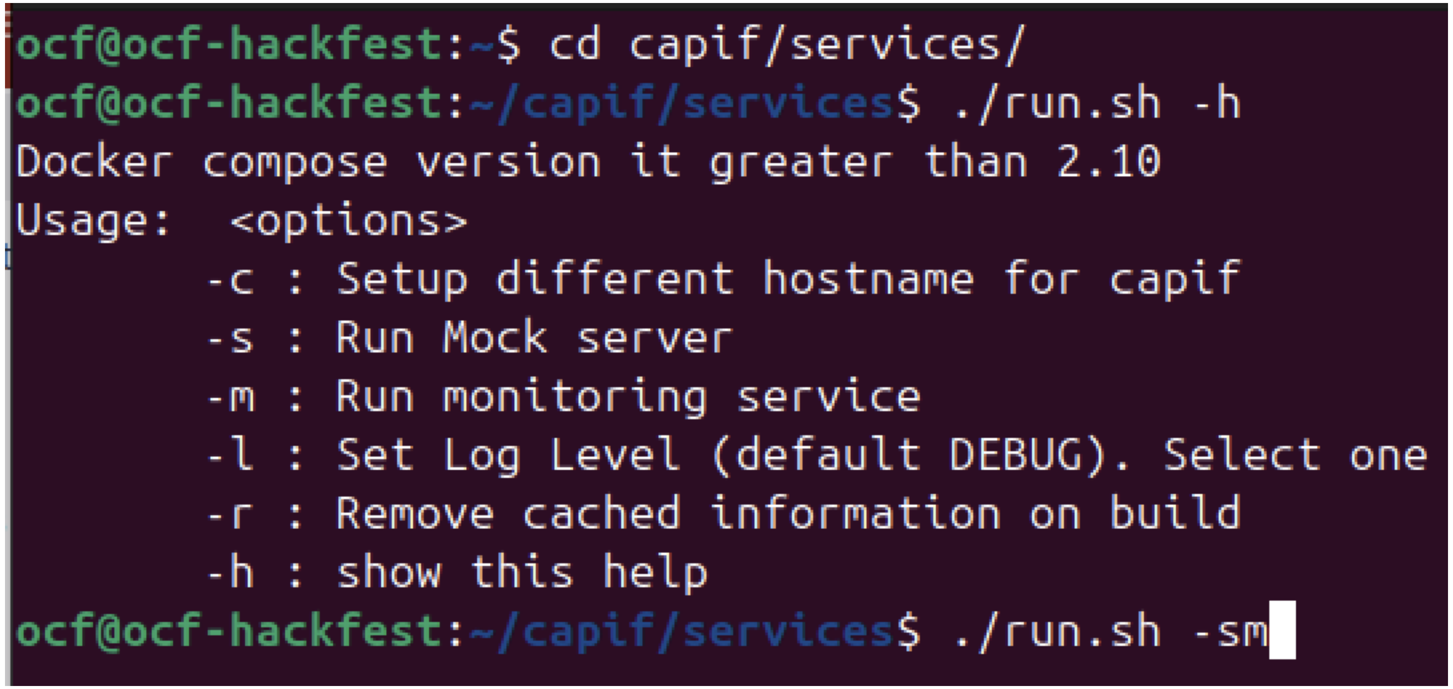
```
git clone --branch staging --single-branch <repository_url>
```

Run Locally OpenCAPIF

Run Locally OpenCAPIF

Go to ~/capif/services directory and execute next commands:

- ./run.sh -h to show help.
- ./run.sh -sm to launch local docker compose.



```
ocf@ocf-hackfest:~$ cd capif/services/
ocf@ocf-hackfest:~/capif/services$ ./run.sh -h
Docker compose version it greater than 2.10
Usage:  <options>
       -c : Setup different hostname for capif
       -s : Run Mock server
       -m : Run monitoring service
       -l : Set Log Level (default DEBUG). Select one
       -r : Remove cached information on build
       -h : show this help
ocf@ocf-hackfest:~/capif/services$ ./run.sh -sm
```

Meanwhile…

Let's take a brief look at the OpenCAPIF components and its architecture.

Architecture

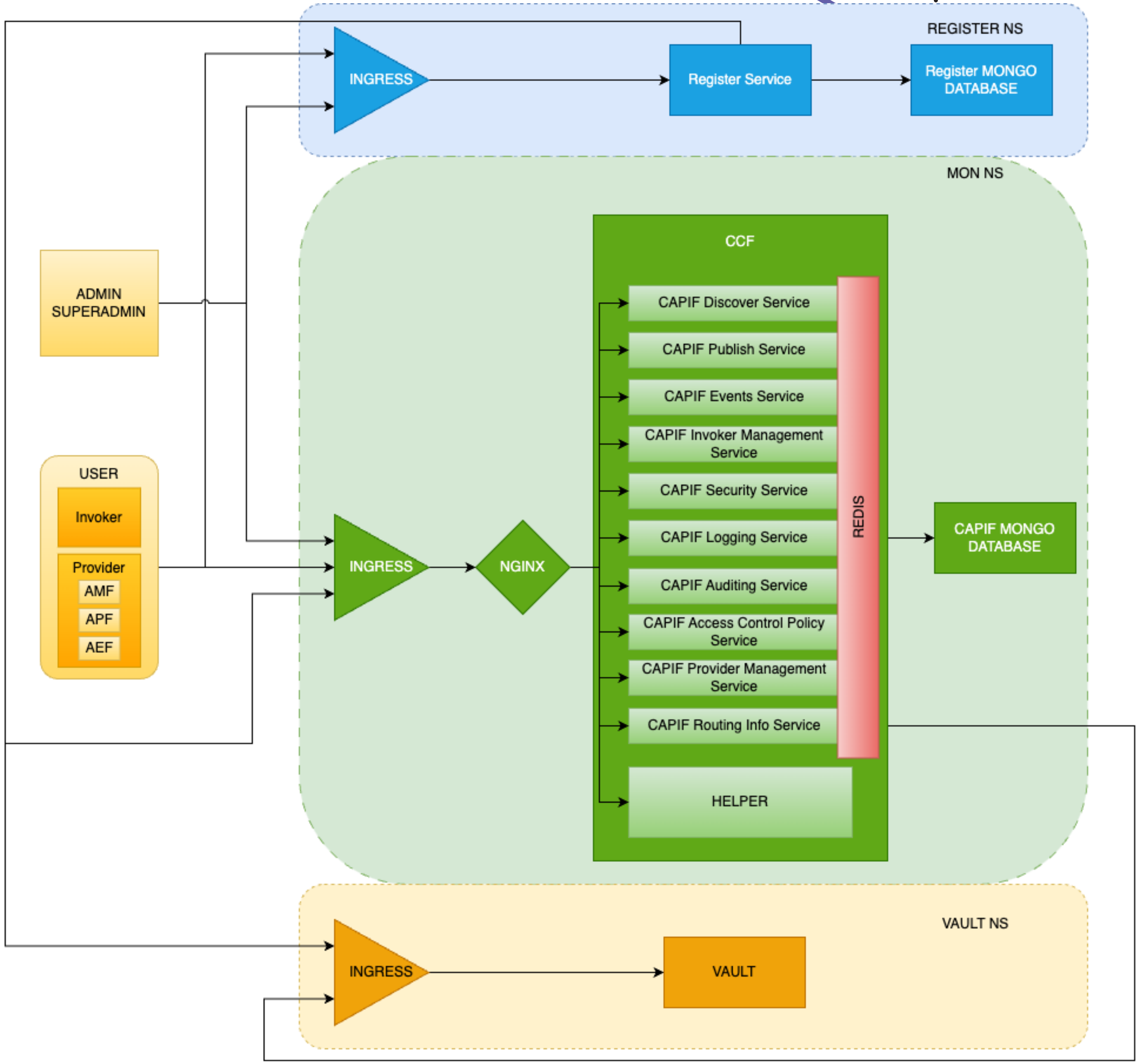
2 types of users:

- Admin/Superadmin
- Invoker/Provider

3 Main Components:

- Register
- CAPIF
- VAULT

All communication between components use Rest APIs



OpenCAPIF

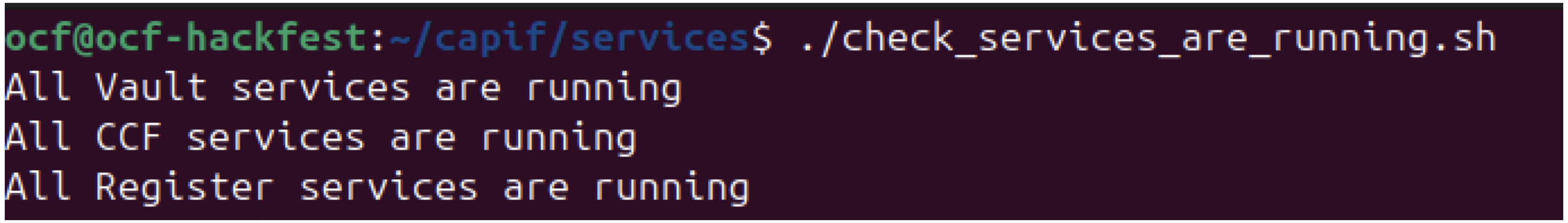
Coffee break and Group Picture
15 minutes

Verify if OpenCAPIF is working

Verify OpenCAPIF is working

Check all docker images are running:

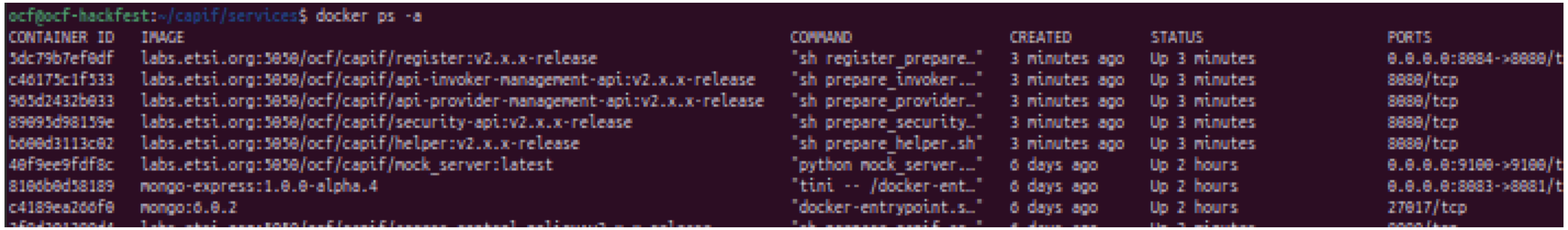
- ./check_services_are_running.sh



```
ocf@ocf-hackfest:~/capif/services$ ./check_services_are_running.sh
All Vault services are running
All CCF services are running
All Register services are running
```

You can also check if all needed docker images are running with command:

- docker ps -a

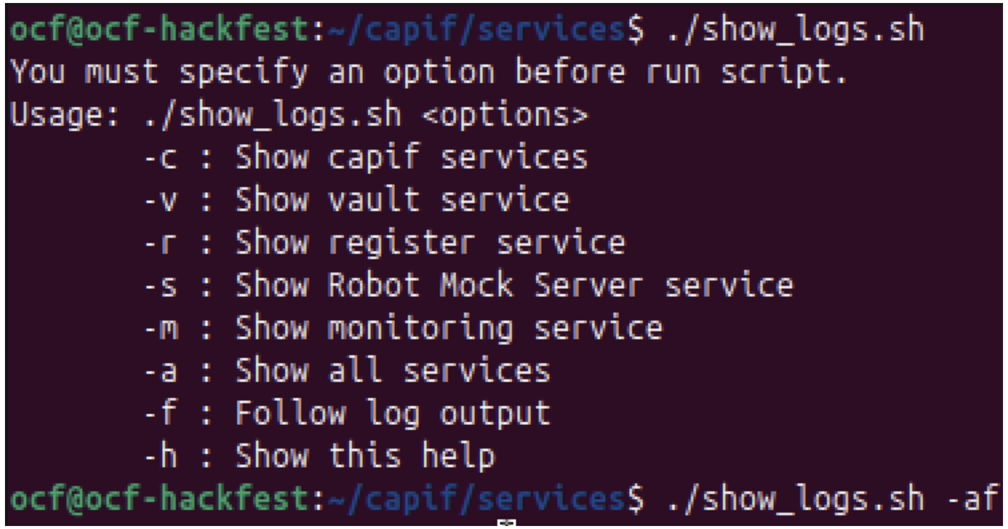


```
ocf@ocf-hackfest:~/capif/services$ docker ps -a
CONTAINER ID   IMAGE                                                                    COMMAND                  CREATED         STATUS          PORTS
5dc79b7ef0df   labs.etsi.org:5050/ocf/capif/register:v2.x.x-release                     "sh register_prepare…"   3 minutes ago   Up 3 minutes    0.0.0.0:8084->8080/t
c40175c1f533   labs.etsi.org:5050/ocf/capif/api-invoker-management-api:v2.x.x-release   "sh prepare_invoker.…"   3 minutes ago   Up 3 minutes    8080/tcp
965d2432b033   labs.etsi.org:5050/ocf/capif/api-provider-management-api:v2.x.x-release  "sh prepare_provider…"   3 minutes ago   Up 3 minutes    8080/tcp
89095d98159e   labs.etsi.org:5050/ocf/capif/security-api:v2.x.x-release                 "sh prepare_security…"   3 minutes ago   Up 3 minutes    8080/tcp
b000d3113c02   labs.etsi.org:5050/ocf/capif/helper:v2.x.x-release                       "sh prepare_helper.sh"   3 minutes ago   Up 3 minutes    8080/tcp
40f9ee9fdf8c   labs.etsi.org:5050/ocf/capif/mock_server:latest                          "python mock_server.…"   6 days ago      Up 2 hours      0.0.0.0:9100->9100/t
8106b0d58189   mongo-express:1.0.0-alpha.4                                              "tini -- /docker-ent…"   6 days ago      Up 2 hours      0.0.0.0:8083->8081/t
c4189ea266f0   mongo:6.0.2                                                              "docker-entrypoint.s…"   6 days ago      Up 2 hours      27017/tcp
```

Verify OpenCAPIF is working

Run show logs:

- ./show_all_logs.sh -af



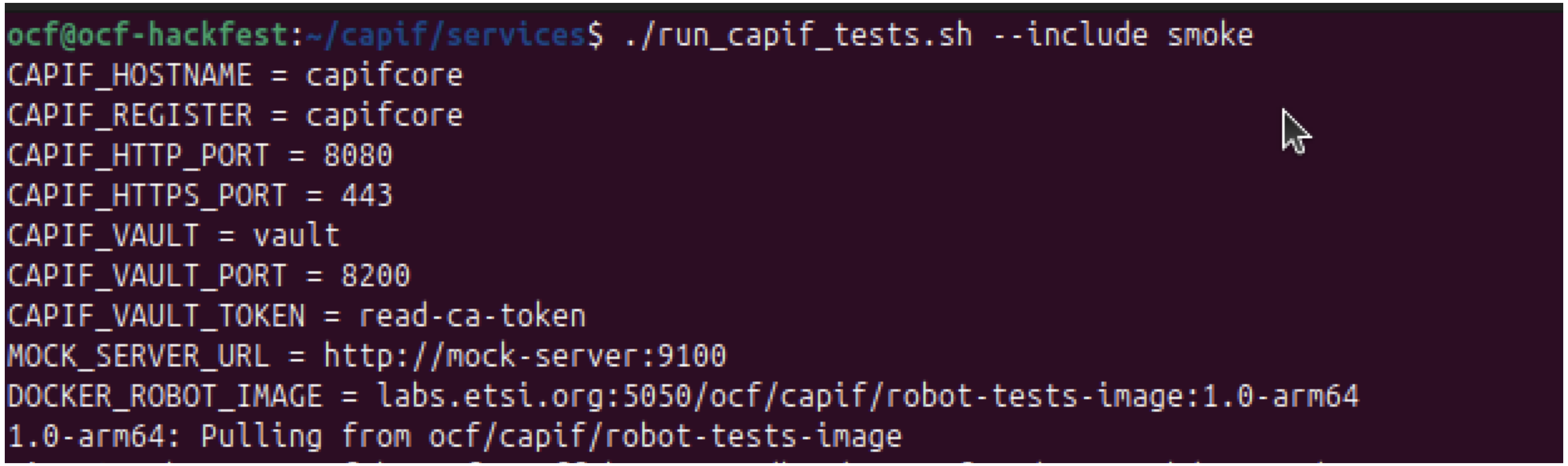
```
ocf@ocf-hackfest:~/capif/services$ ./show_logs.sh
You must specify an option before run script.
Usage: ./show_logs.sh <options>
        -c : Show capif services
        -v : Show vault service
        -r : Show register service
        -s : Show Robot Mock Server service
        -m : Show monitoring service
        -a : Show all services
        -f : Follow log output
        -h : Show this help
ocf@ocf-hackfest:~/capif/services$ ./show_logs.sh -af
```

Verify OpenCAPIF is working

OpenCAPIF

Open new tab in terminal and execute robot smoke tests:

- ./run_capif_tests.sh --include smoke

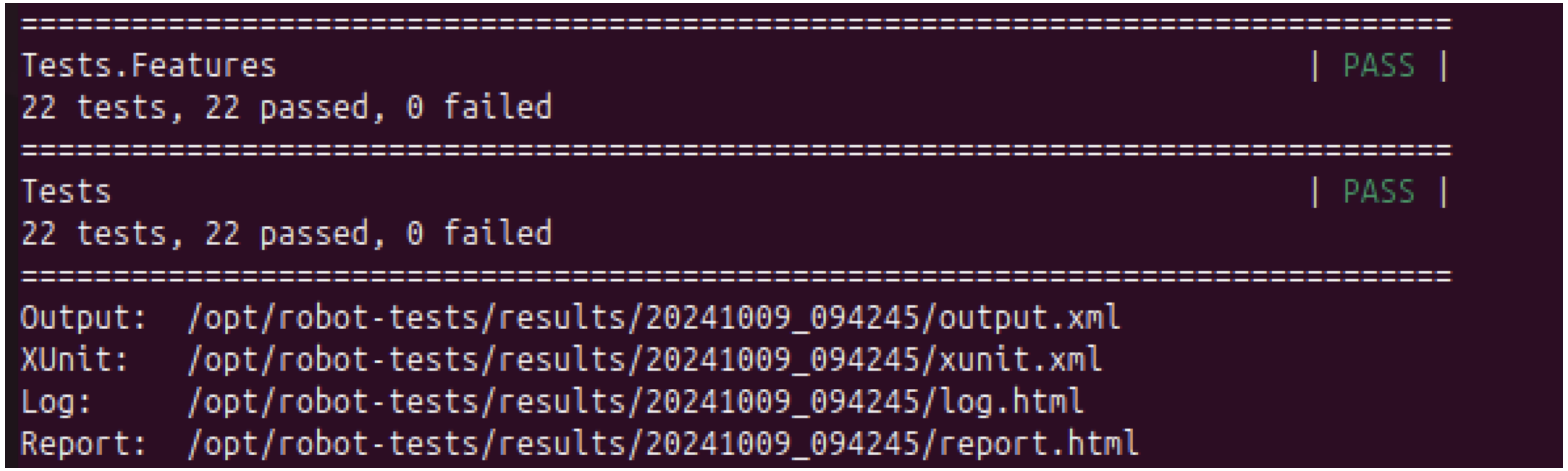


```
ocf@ocf-hackfest:~/capif/services$ ./run_capif_tests.sh --include smoke
CAPIF_HOSTNAME = capifcore
CAPIF_REGISTER = capifcore
CAPIF_HTTP_PORT = 8080
CAPIF_HTTPS_PORT = 443
CAPIF_VAULT = vault
CAPIF_VAULT_PORT = 8200
CAPIF_VAULT_TOKEN = read-ca-token
MOCK_SERVER_URL = http://mock-server:9100
DOCKER_ROBOT_IMAGE = labs.etsi.org:5050/ocf/capif/robot-tests-image:1.0-arm64
1.0-arm64: Pulling from ocf/capif/robot-tests-image
```

This will download Robot Framework image and execute smoke tagged tests

Verify OpenCAPIF is working

If all tests were PASSED, the OpenCAPIF is working fine:



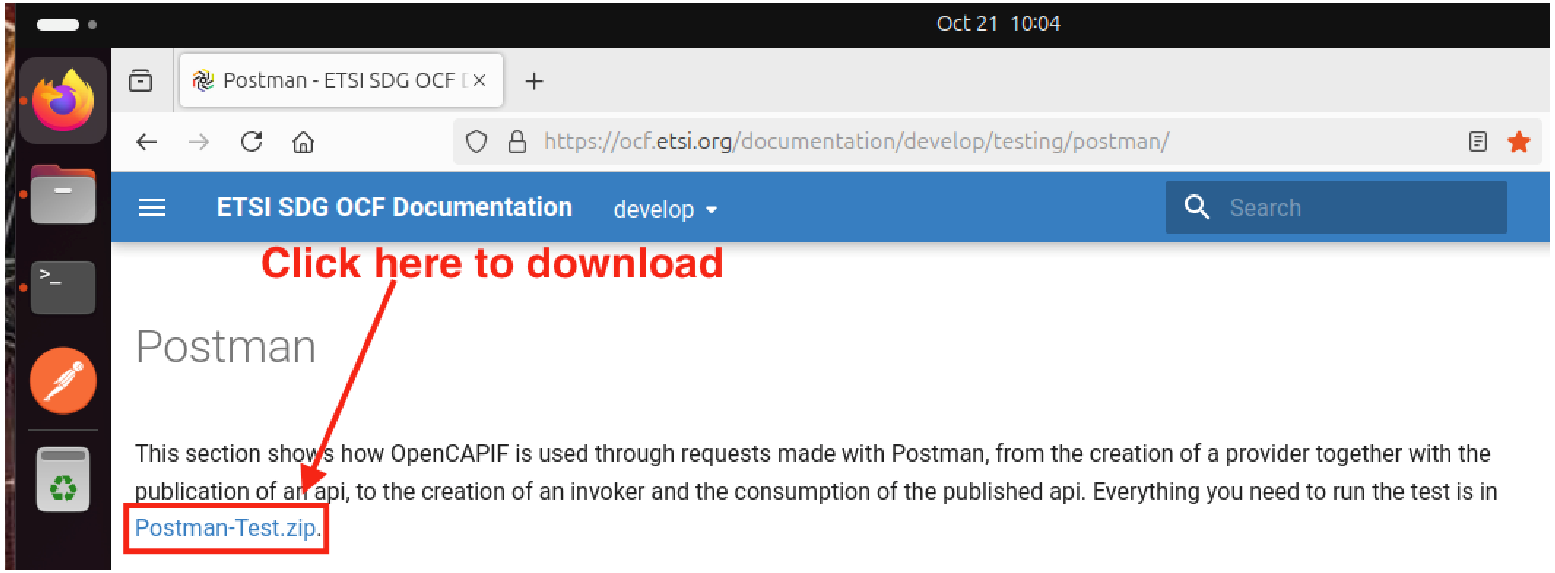
```
==============================================================================
Tests.Features                                                        | PASS |
22 tests, 22 passed, 0 failed
==============================================================================
Tests                                                                 | PASS |
22 tests, 22 passed, 0 failed
==============================================================================
Output:  /opt/robot-tests/results/20241009_094245/output.xml
XUnit:   /opt/robot-tests/results/20241009_094245/xunit.xml
Log:     /opt/robot-tests/results/20241009_094245/log.html
Report:  /opt/robot-tests/results/20241009_094245/report.html
```

Now we are ready to start !!!

Setup Postman

Download Postman collection

First step is download Postman Collection from OCF web page:



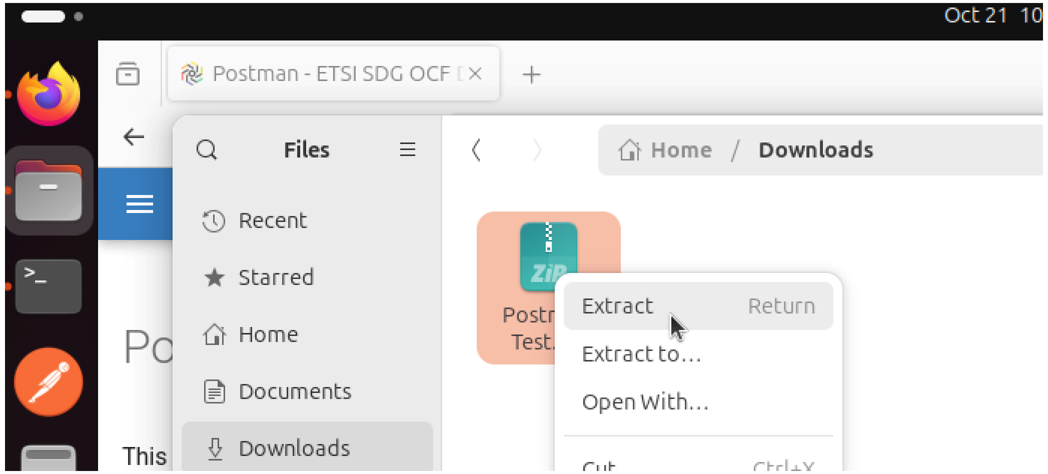
Oct 21 10:04
Postman - ETSI SDG OCF
https://ocf.etsi.org/documentation/develop/testing/postman/
ETSI SDG OCF Documentation
develop
Search
Click here to download

Postman

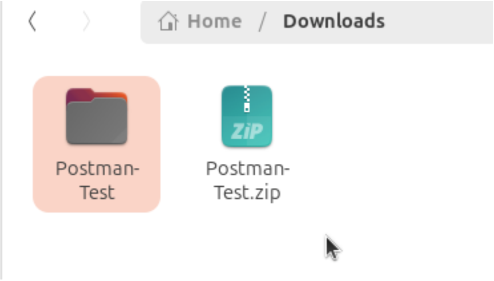
This section shows how OpenCAPIF is used through requests made with Postman, from the creation of a provider together with the publication of an api, to the creation of an invoker and the consumption of the published api. Everything you need to run the test is in Postman-Test.zip.

Extract Postman collection

Extract all files inside zip file:

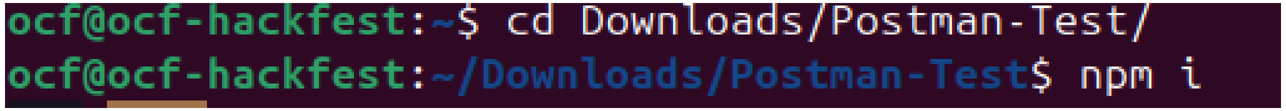


You will see this:

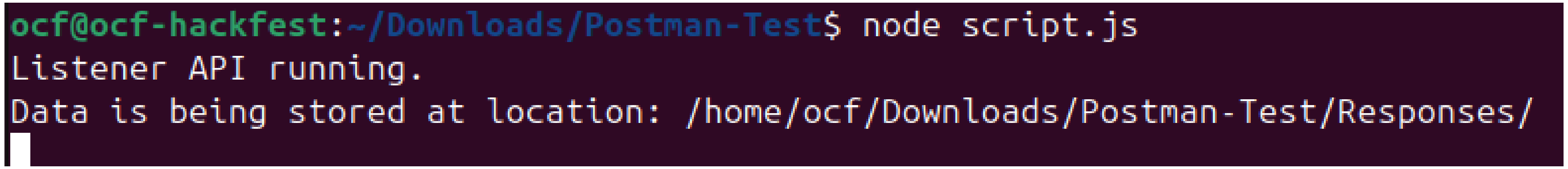


Open Terminal and go to folder

Go to folder and execute npm i and run node script.js

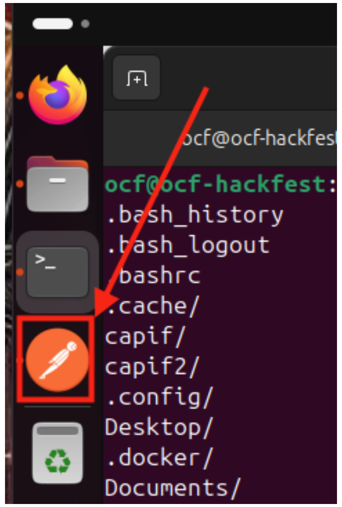


```
ocf@ocf-hackfest:~$ cd Downloads/Postman-Test/
ocf@ocf-hackfest:~/Downloads/Postman-Test$ npm i
```



```
ocf@ocf-hackfest:~/Downloads/Postman-Test$ node script.js
Listener API running.
Data is being stored at location: /home/ocf/Downloads/Postman-Test/Responses/
```

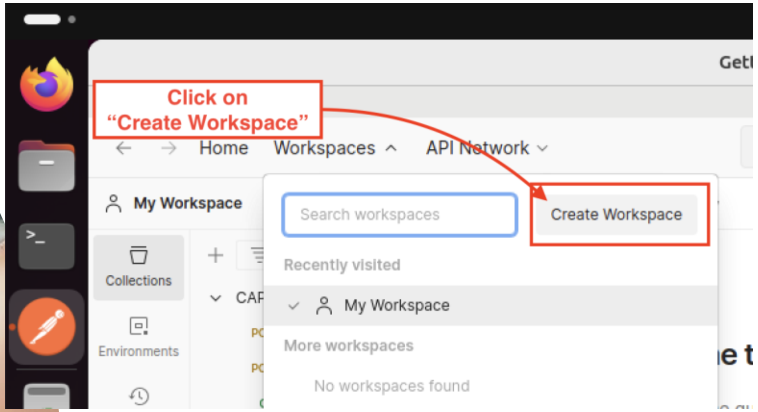
Open Postman:

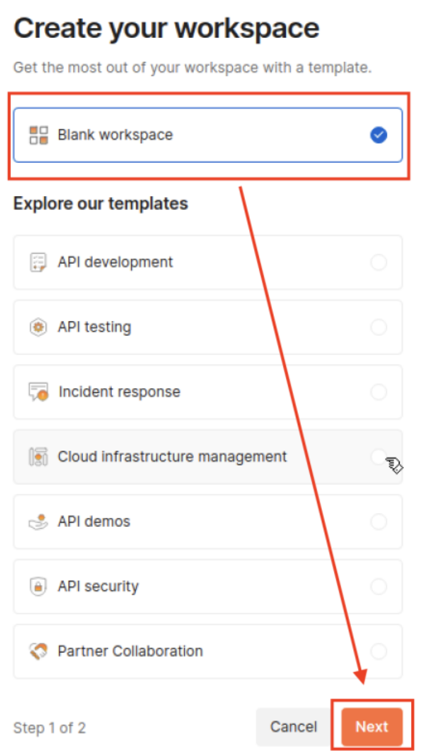


```
ocf@ocf-hackfes
ocf@ocf-hackfest:
.bash_history
.bash_logout
.bashrc
.cache/
capif/
capif2/
.config/
Desktop/
.docker/
Documents/
```

Create a Workspace

Click on “Create Workspace”:





Create your workspace

Get the most out of your workspace with a template.

● Blank workspace

Explore our templates

○ API development

○ API testing

○ Incident response

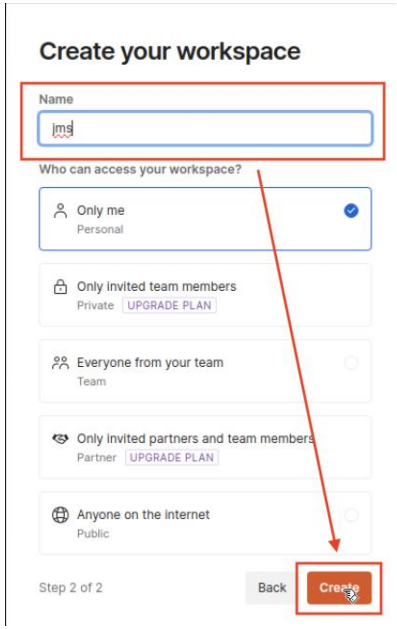
○ Cloud infrastructure management

○ API demos

○ API security

○ Partner Collaboration

Step 1 of 2



Create your workspace

Name: jms

Who can access your workspace?

● Only me
Personal

○ Only invited team members
Private UPGRADE PLAN

○ Everyone from your team
Team

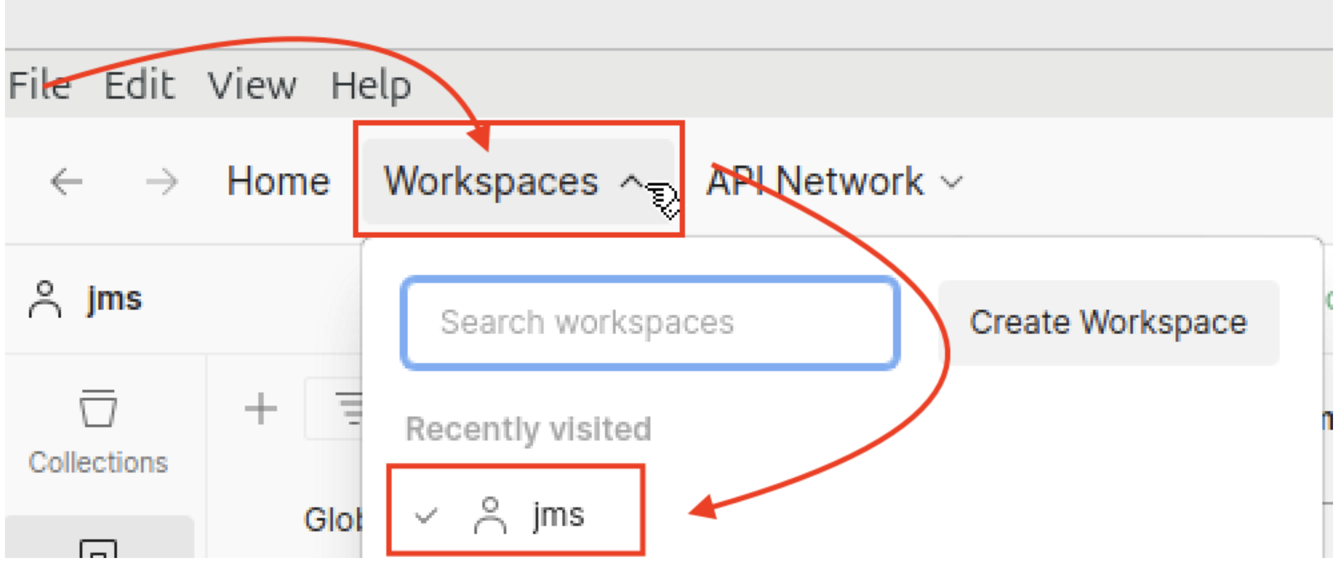
○ Only invited partners and team members
Partner UPGRADE PLAN

○ Anyone on the internet
Public

Step 2 of 2

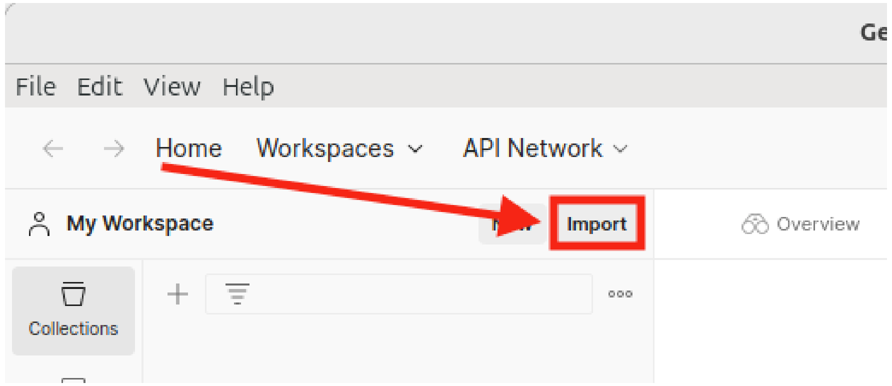
Select your Workspace

Click on “Workspaces” and select the created Workspace:

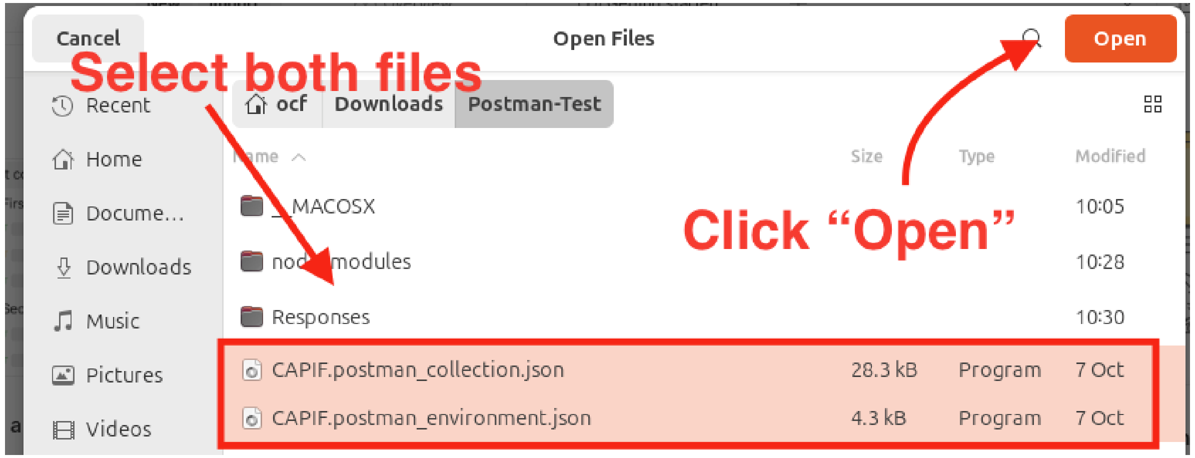


Import Postman collection

Click on import:



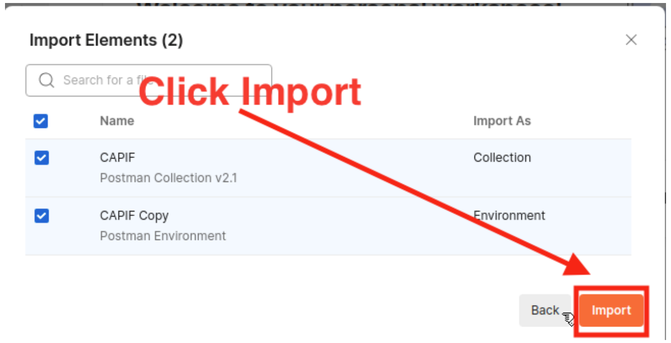
Select both files CAPIF collection an environment and click "Open"



Name	Size	Type	Modified
_MACOSX			10:05
node_modules			10:28
Responses			10:30
CAPIF.postman_collection.json	28.3 kB	Program	7 Oct
CAPIF.postman_environment.json	4.3 kB	Program	7 Oct

Finish import and select Env

Click on import:

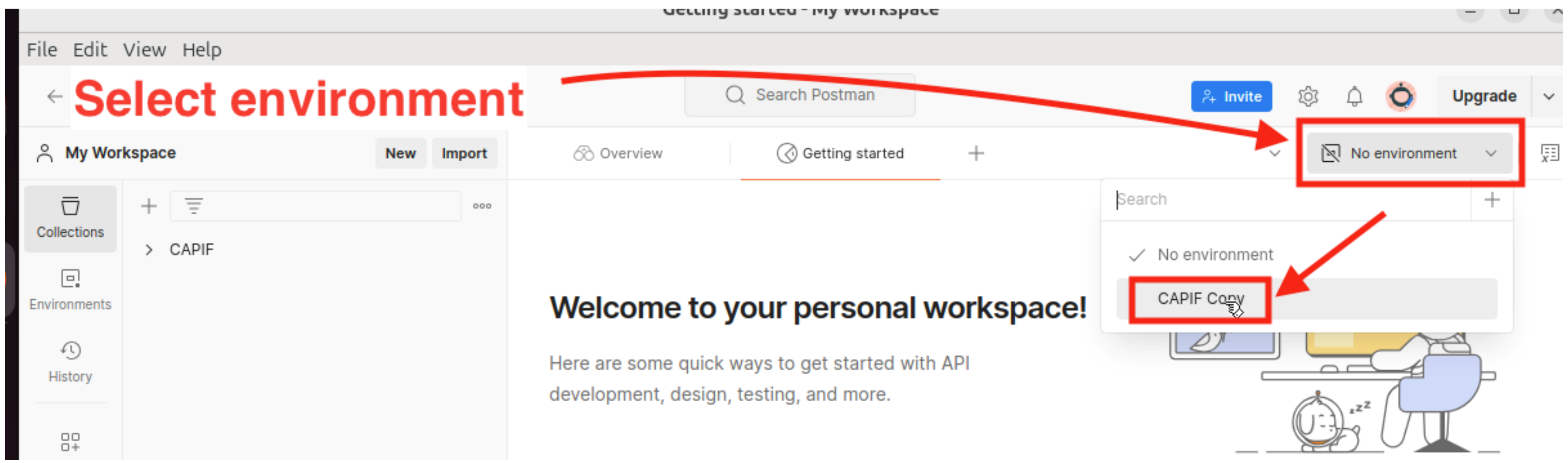


Import Elements (2)

Click Import

☑	Name	Import As
☑	CAPIF Postman Collection v2.1	Collection
☑	CAPIF Copy Postman Environment	Environment

Select CAPIF Environment:



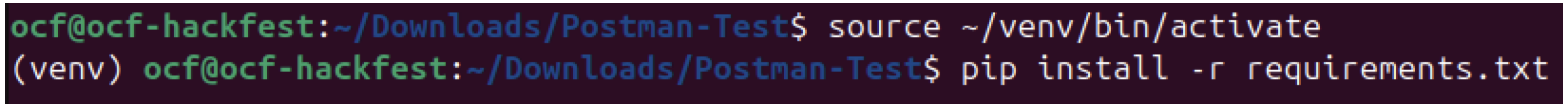
Select environment

Welcome to your personal workspace!

Here are some quick ways to get started with API development, design, testing, and more.

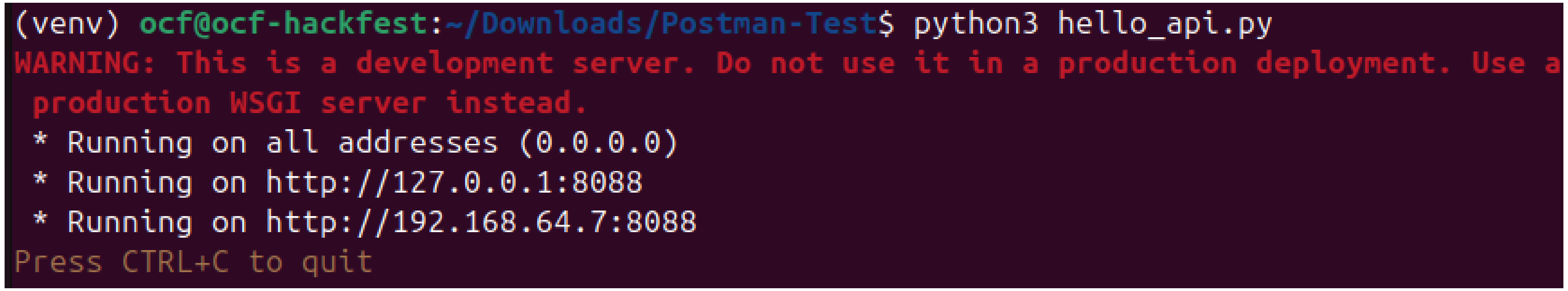
Install Python dependencies

Click on new tag at terminal and run:



```
ocf@ocf-hackfest:~/Downloads/Postman-Test$ source ~/venv/bin/activate
(venv) ocf@ocf-hackfest:~/Downloads/Postman-Test$ pip install -r requirements.txt
```

And we can run hello_api client.

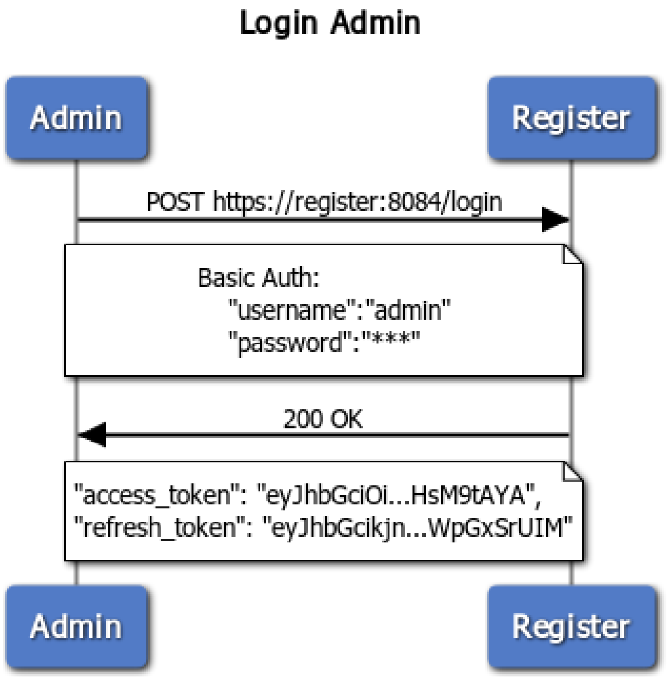


```
(venv) ocf@ocf-hackfest:~/Downloads/Postman-Test$ python3 hello_api.py
WARNING: This is a development server. Do not use it in a production deployment. Use a
 production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8088
 * Running on http://192.168.64.7:8088
Press CTRL+C to quit
```

User Registration Flow by Administrator

Login as Admin

First step to act as Admin is login in order to get Access token from Register Service:

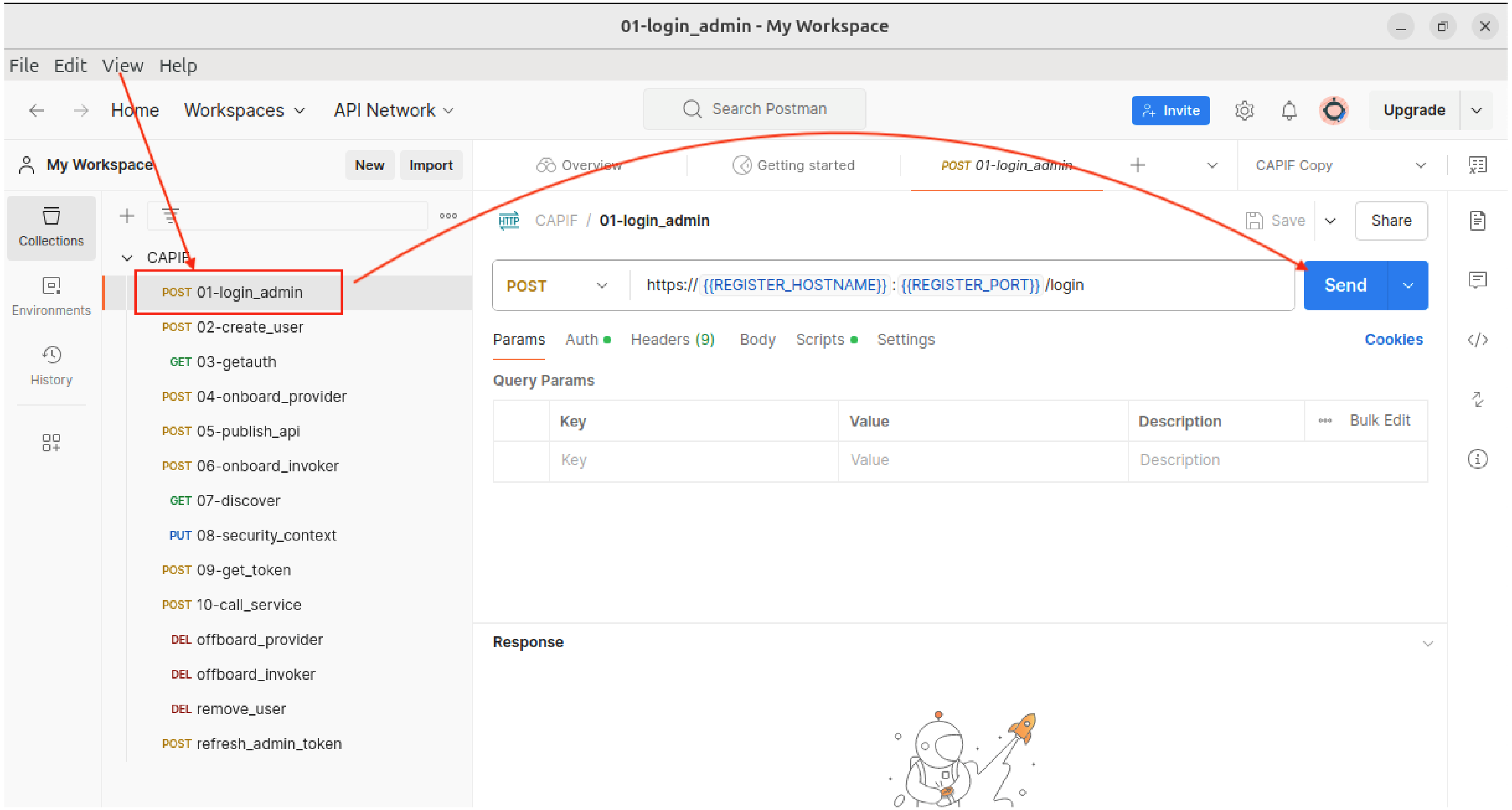


```
Basic Auth:
    "username":"admin"
    "password":"***"
```

```
"access_token": "eyJhbGciOi...HsM9tAYA",
"refresh_token": "eyJhbGcikjn...WpGxSrUIM"
```

Login as Admin

Select 01-login_admin request and click on Send:



01-login_admin - My Workspace

File Edit View Help

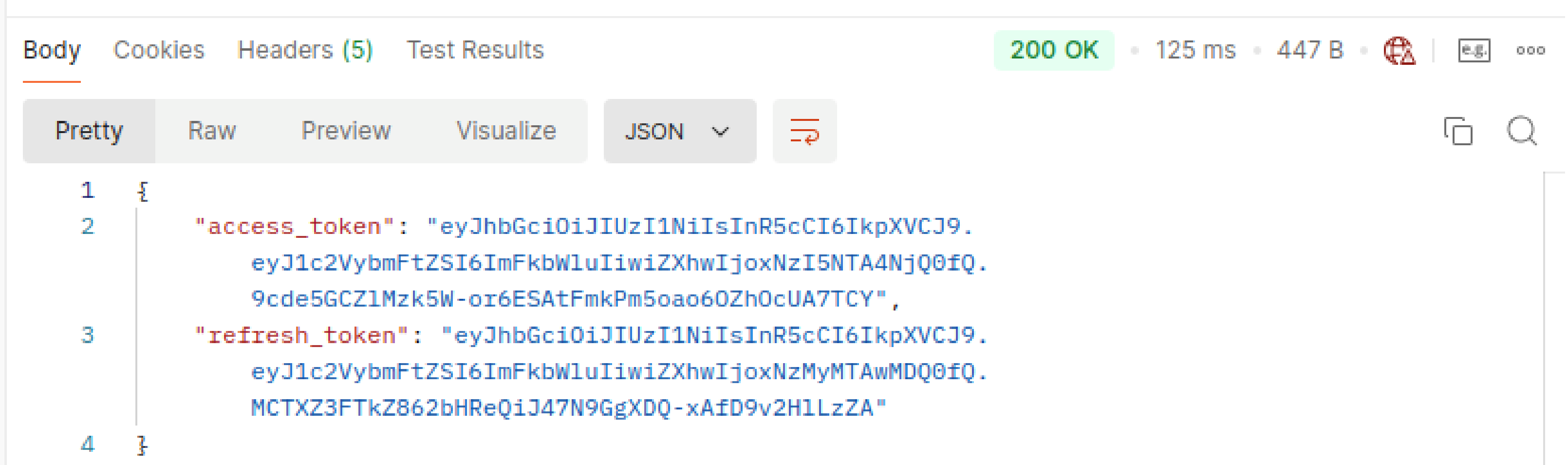
CAPIF / 01-login_admin

POST https://{{REGISTER_HOSTNAME}}:{{REGISTER_PORT}}/login

Send

Login as Admin

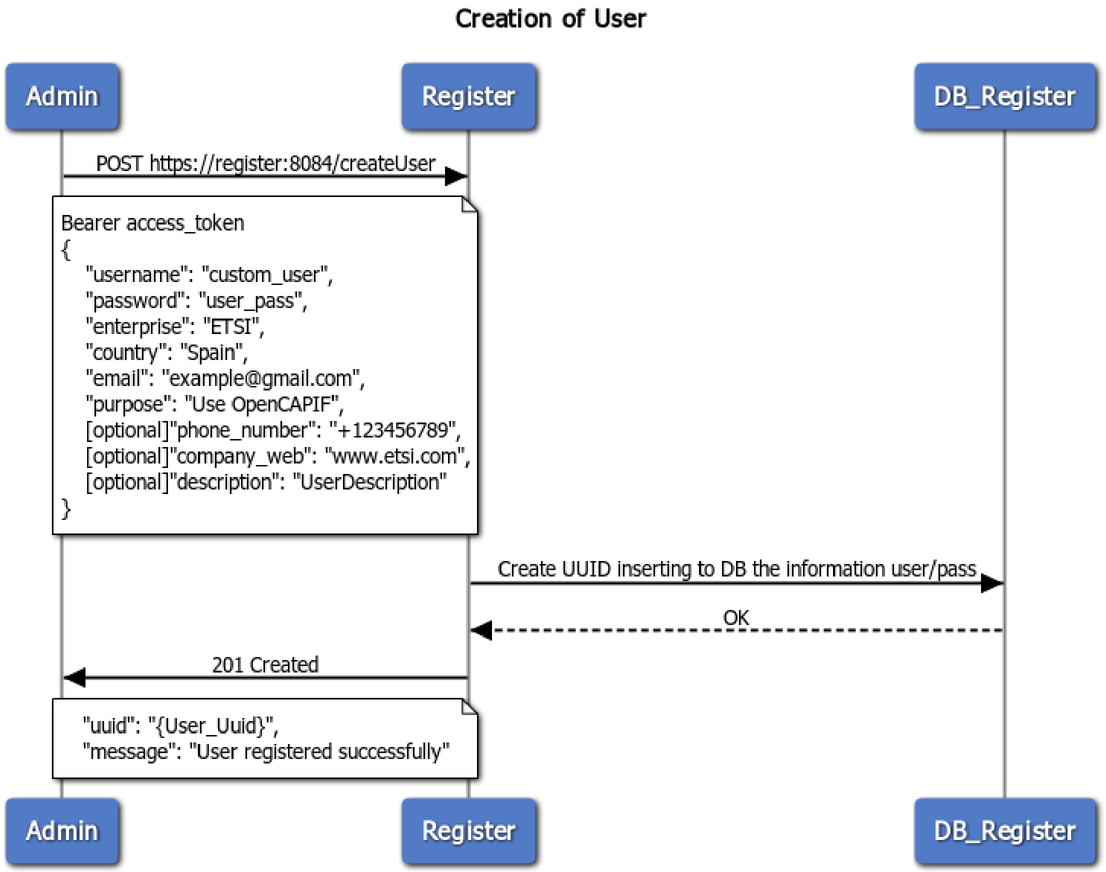
We Will get 200 OK Response with Access token:



```
{
    "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VybmFtZSI6ImFkbWluIiwiZXhwIjoxNzI5NTA4NjQ0fQ.
        9cde5GCZlMzk5W-or6ESAtFmkPm5oao6OZhOcUA7TCY",
    "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VybmFtZSI6ImFkbWluIiwiZXhwIjoxNzMyMTAwMDQ0fQ.
        MCTXZ3FTkZ862bHReQiJ47N9GgXDQ-xAfD9v2HlLzZA"
}
```

Create new user as admin

After login as administrator, new user can be created:



Creation of User

Admin
Register
DB_Register
POST https://register:8084/createUser

```
Bearer access_token
{
   "username": "custom_user",
   "password": "user_pass",
   "enterprise": "ETSI",
   "country": "Spain",
   "email": "example@gmail.com",
   "purpose": "Use OpenCAPIF",
   [optional]"phone_number": "+123456789",
   [optional]"company_web": "www.etsi.com",
   [optional]"description": "UserDescription"
}
```

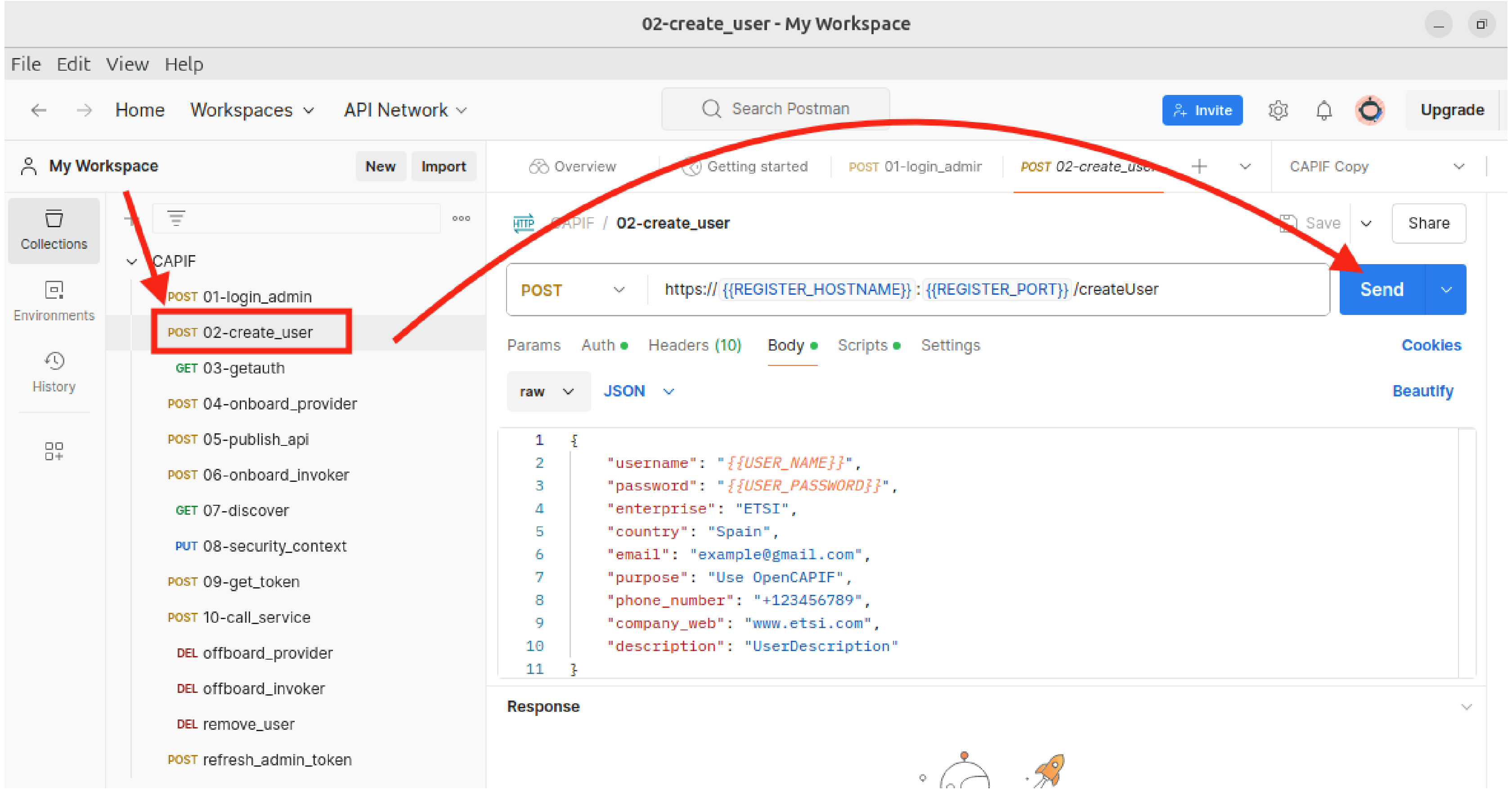
Create UUID inserting to DB the information user/pass
OK
201 Created

```
   "uuid": "{User_Uuid}",
   "message": "User registered successfully"
```

Admin
Register
DB_Register

Create new user as admin

Select 02-create_user request and click on “Send”:

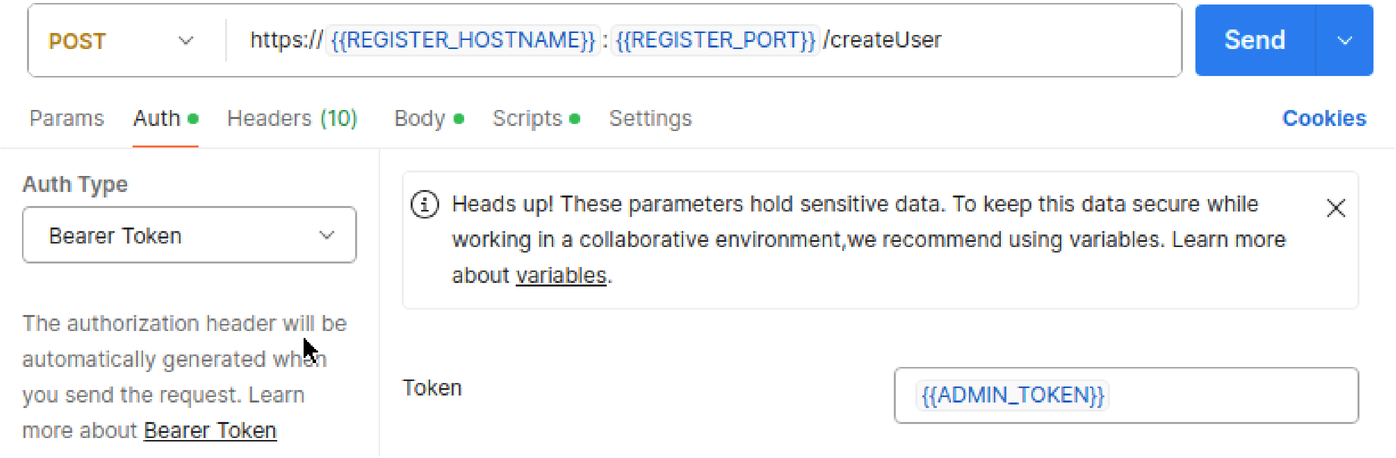


POST https://{{REGISTER_HOSTNAME}}:{{REGISTER_PORT}}/createUser

```
{
    "username": "{{USER_NAME}}",
    "password": "{{USER_PASSWORD}}",
    "enterprise": "ETSI",
    "country": "Spain",
    "email": "example@gmail.com",
    "purpose": "Use OpenCAPIF",
    "phone_number": "+123456789",
    "company_web": "www.etsi.com",
    "description": "UserDescription"
}
```

Create new user as admin

We can check Auth tag to see token used:



POST https://{{REGISTER_HOSTNAME}}:{{REGISTER_PORT}}/createUser

Auth Type Bearer Token

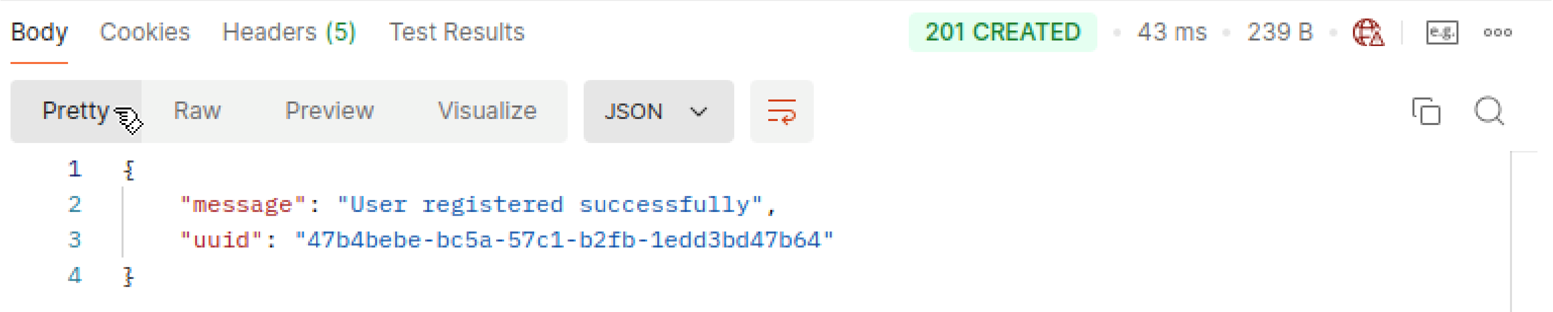
The authorization header will be automatically generated when you send the request. Learn more about Bearer Token

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

Token {{ADMIN_TOKEN}}

Create new user as admin

We will get next 201 Created response with next body:



```
{
    "message": "User registered successfully",
    "uuid": "47b4bebe-bc5a-57c1-b2fb-1edd3bd47b64"
}
```

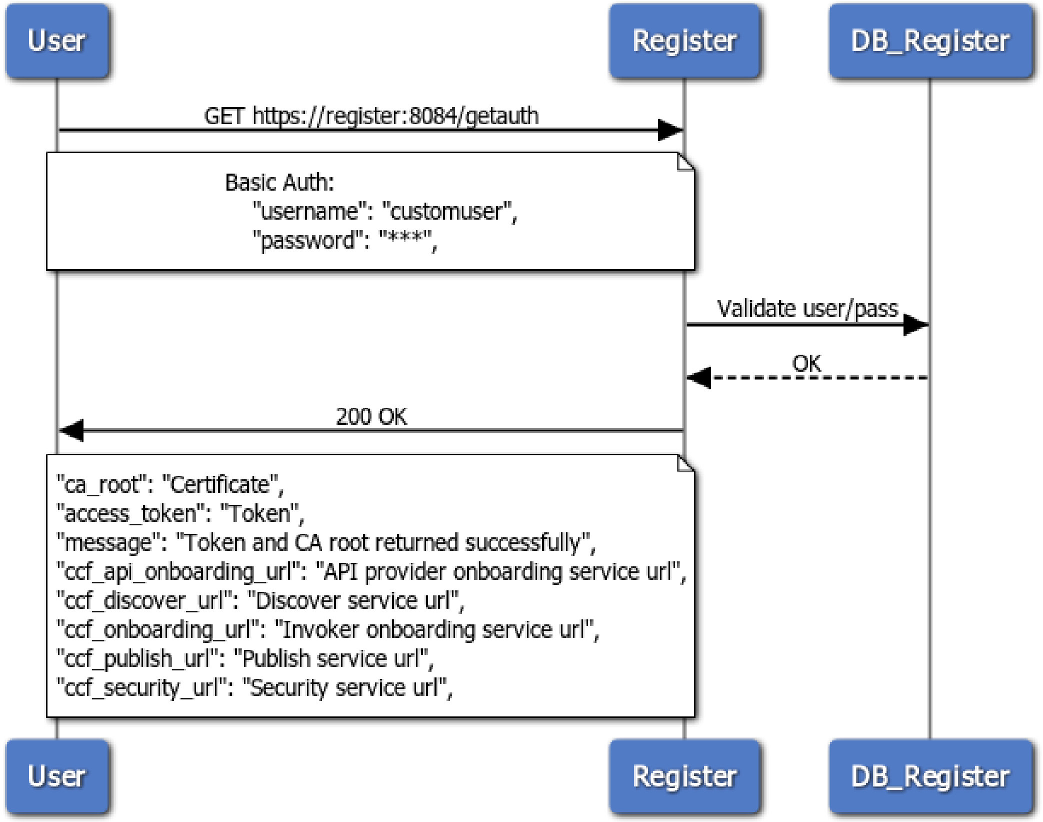
Now we have a user created by Admin in our local OpenCAPIF.

Provider Onboarding flow by customer User

Get token to interact with CCF

First is obtain Access token and endpoints for user:

User Getauth

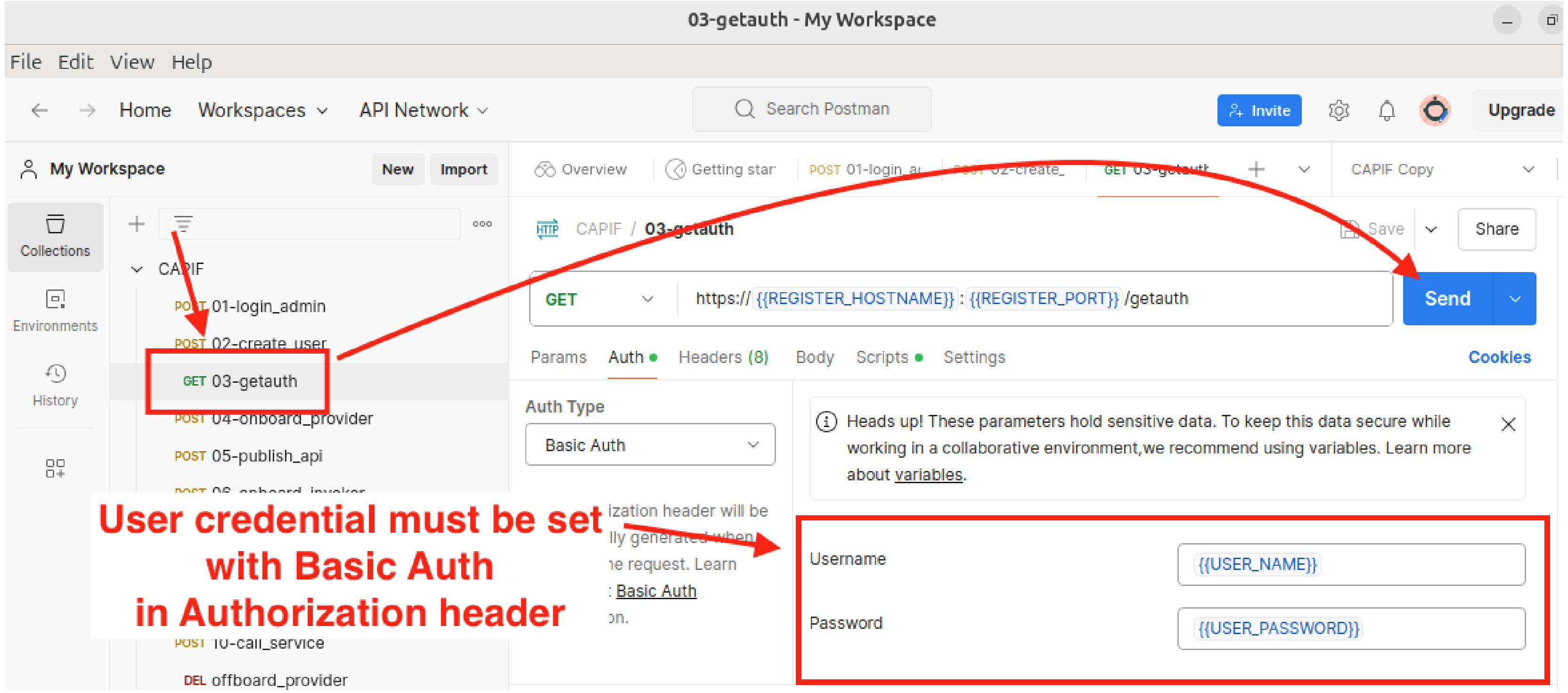


```
Basic Auth:
    "username": "customuser",
    "password": "***",
```

```
"ca_root": "Certificate",
"access_token": "Token",
"message": "Token and CA root returned successfully",
"ccf_api_onboarding_url": "API provider onboarding service url",
"ccf_discover_url": "Discover service url",
"ccf_onboarding_url": "Invoker onboarding service url",
"ccf_publish_url": "Publish service url",
"ccf_security_url": "Security service url",
```

Get token to interact with CCF

Select request 03-getauth and click on "Send":



GET https:// {{REGISTER_HOSTNAME}} : {{REGISTER_PORT}} /getauth

Auth Type
Basic Auth

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

ization header will be lly generated when e request. Learn : Basic Auth on.

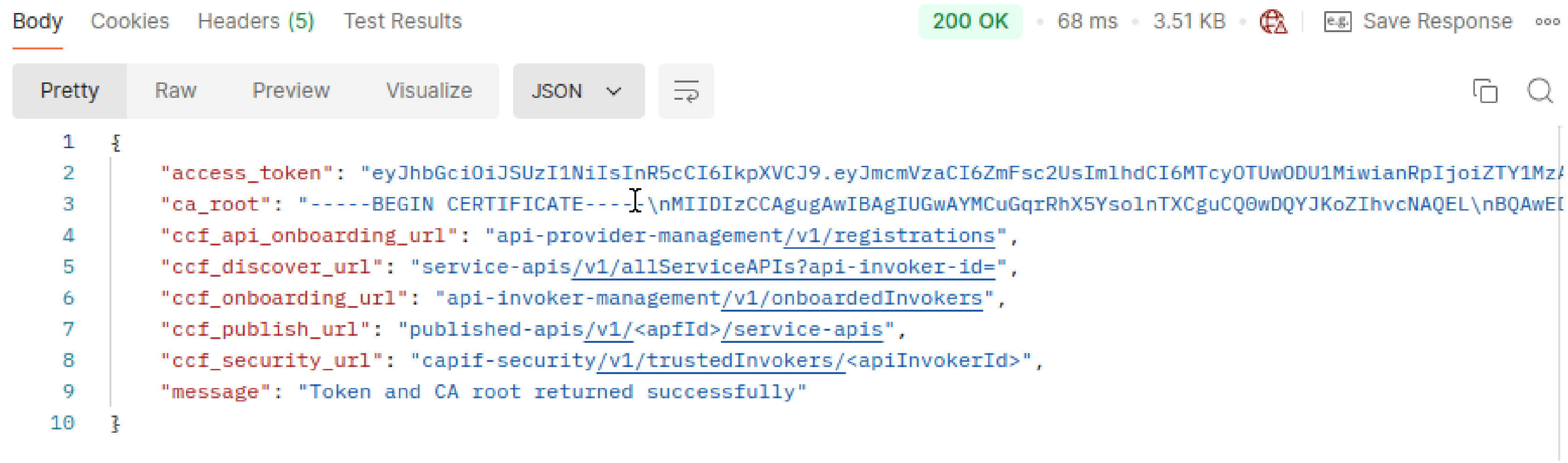
Username {{USER_NAME}}

Password {{USER_PASSWORD}}

User credential must be set with Basic Auth in Authorization header

Get token to interact with CCF

We will see 200 OK response with next body:



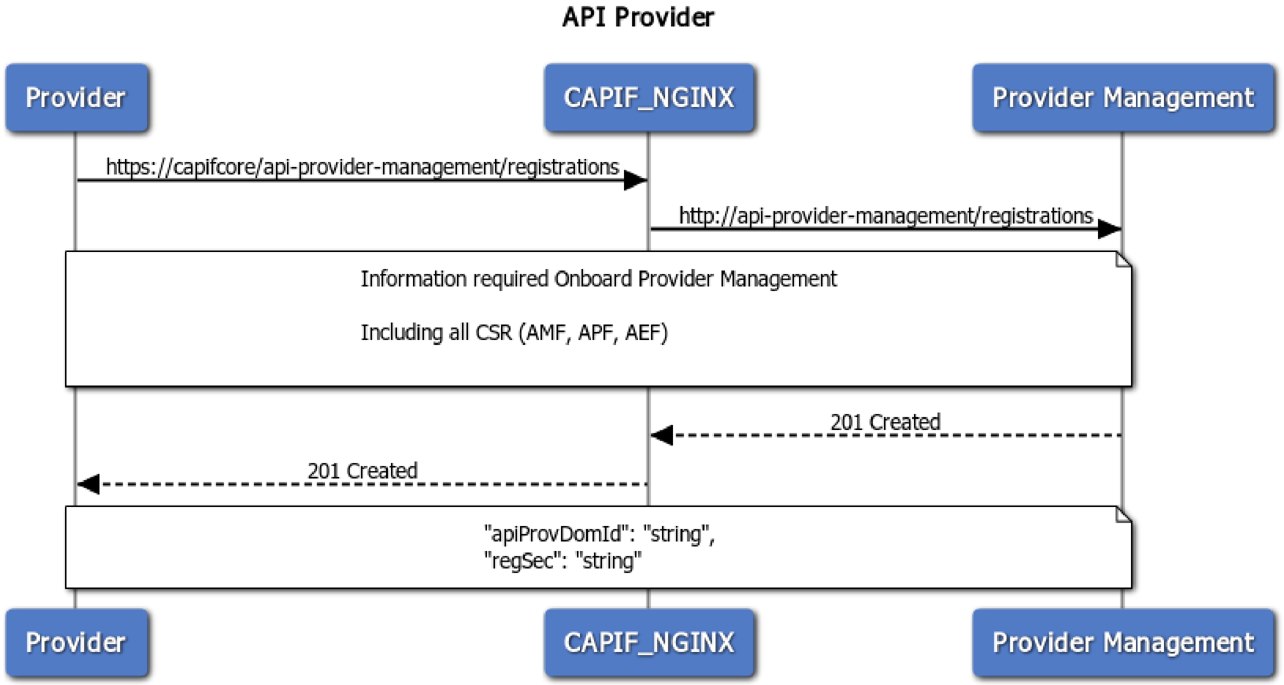
```
{
    "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2UsImlhdCI6MTcyOTUwODU1MiwianRpIjoiZTY1Mz
    "ca_root": "-----BEGIN CERTIFICATE-----\nMIIDIzCCAgugAwIBAgIUGwAYMCuGqrRhX5YsolnTXCguCQ0wDQYJKoZIhvcNAQEL\nBQAwEI
    "ccf_api_onboarding_url": "api-provider-management/v1/registrations",
    "ccf_discover_url": "service-apis/v1/allServiceAPIs?api-invoker-id=",
    "ccf_onboarding_url": "api-invoker-management/v1/onboardedInvokers",
    "ccf_publish_url": "published-apis/v1/<apfId>/service-apis",
    "ccf_security_url": "capif-security/v1/trustedInvokers/<apiInvokerId>",
    "message": "Token and CA root returned successfully"
}
```

We must use temporally **access_token** to interact with CCF as client.

Also in this response we can see urls to send each request, like onboarding, publish, discover,...

Onboard a provider

Now we can onboard the provider:



API Provider

Provider

CAPIF_NGINX

Provider Management

https://capifcore/api-provider-management/registrations

http://api-provider-management/registrations

Information required Onboard Provider Management

Including all CSR (AMF, APF, AEF)

201 Created

201 Created

"apiProvDomId": "string",
"regSec": "string"

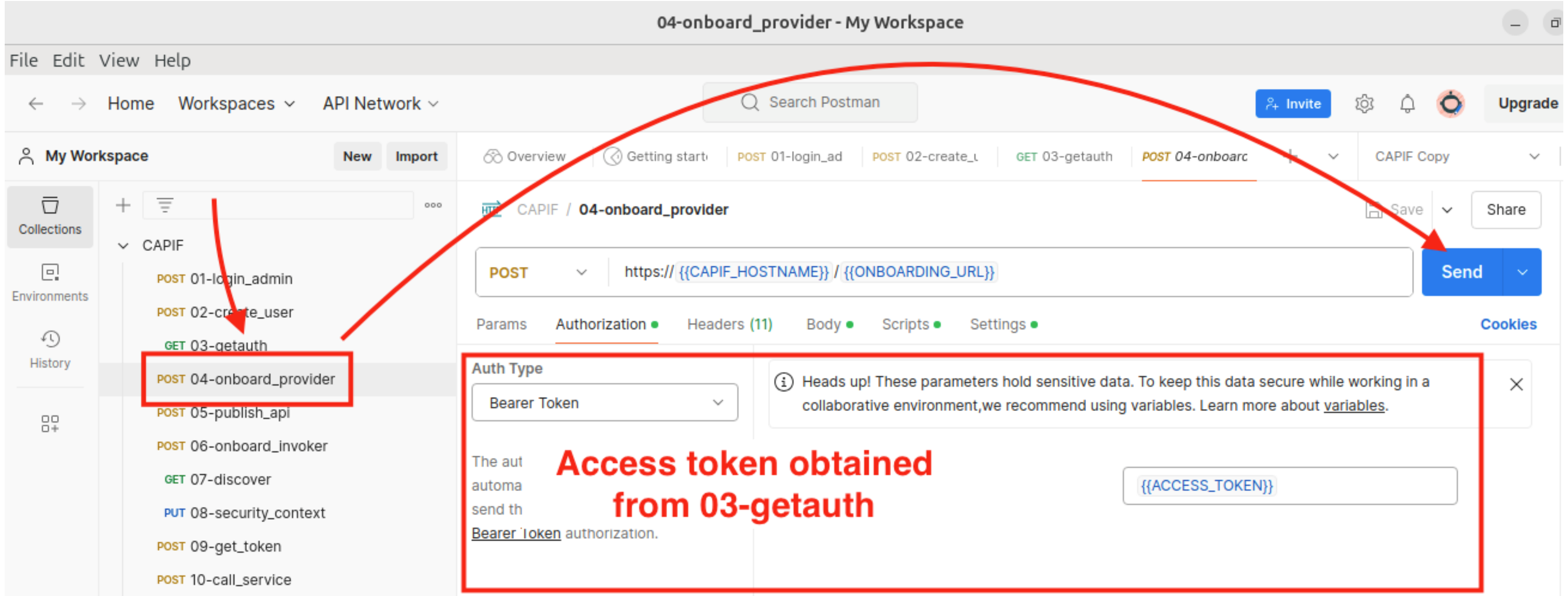
Provider

CAPIF_NGINX

Provider Management

Onboard a provider

Select 04-onboard_provider and click “Send”:



POST https://{{CAPIF_HOSTNAME}}/{{ONBOARDING_URL}}

Auth Type: Bearer Token

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

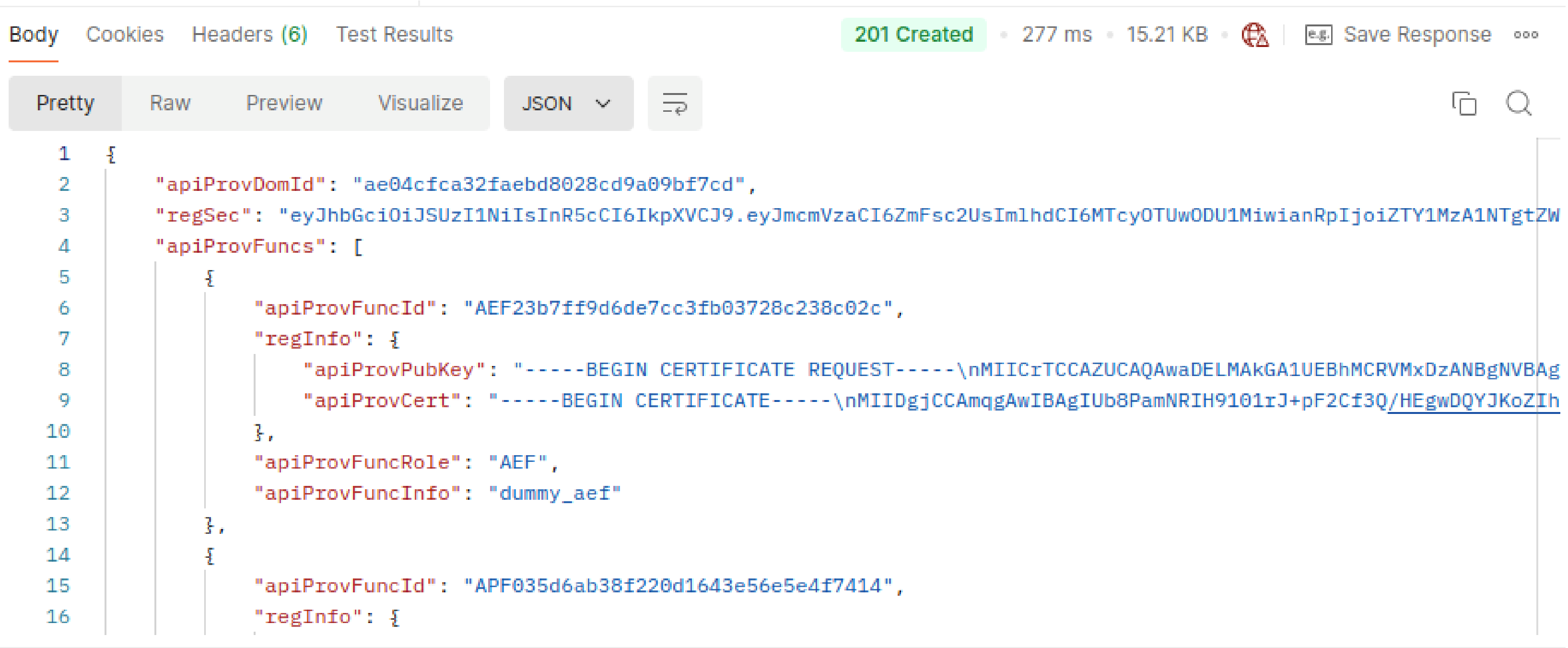
The aut
automa
send th
Bearer Token authorization.

{{ACCESS_TOKEN}}

Access token obtained from 03-getauth

Onboard a provider

Response received will be 201 Created with certificates signed and apiProvDomId:

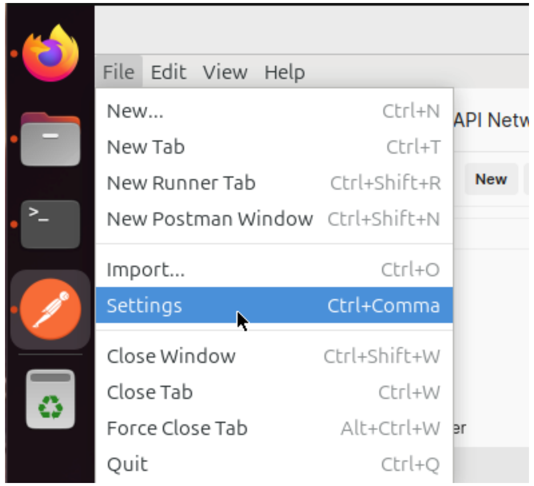


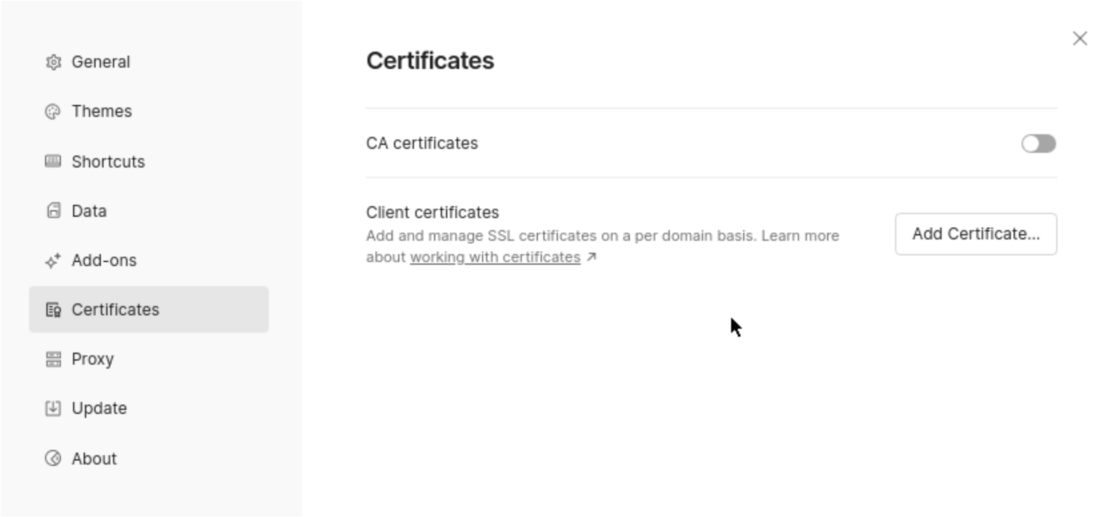
```
{
    "apiProvDomId": "ae04cfca32faebd8028cd9a09bf7cd",
    "regSec": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2UsImlhdCI6MTcyOTUwODU1MiwianRpIjoiZTY1MzA1NTgtZW
    "apiProvFuncs": [
        {
            "apiProvFuncId": "AEF23b7ff9d6de7cc3fb03728c238c02c",
            "regInfo": {
                "apiProvPubKey": "-----BEGIN CERTIFICATE REQUEST-----\nMIICrTCCAZUCAQAwaDELMAkGA1UEBhMCRVMxDzANBgNVBAg
                "apiProvCert": "-----BEGIN CERTIFICATE-----\nMIIDgjCCAmqgAwIBAgIUb8PamNRIH9101rJ+pF2Cf3Q/HEgwDQYJKoZIh
            },
            "apiProvFuncRole": "AEF",
            "apiProvFuncInfo": "dummy_aef"
        },
        {
            "apiProvFuncId": "APF035d6ab38f220d1643e56e5e4f7414",
            "regInfo": {
```

Configure Certificates in Postman

Before go to next request, we will need to configure Postman with credentials obtained, all is explained in documentation.

Go to Settings and open certificates section:





Certificates

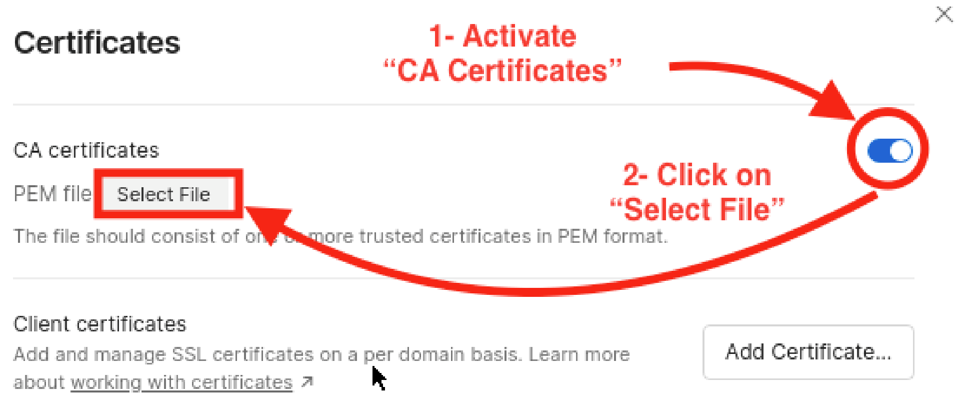
CA certificates

Client certificates
Add and manage SSL certificates on a per domain basis. Learn more about working with certificates ↗

Add Certificate...

Configure Certificates in Postman

Activate CA certificates and select ca_cert.pem:



1- Activate "CA Certificates"

2- Click on "Select File"

Certificates

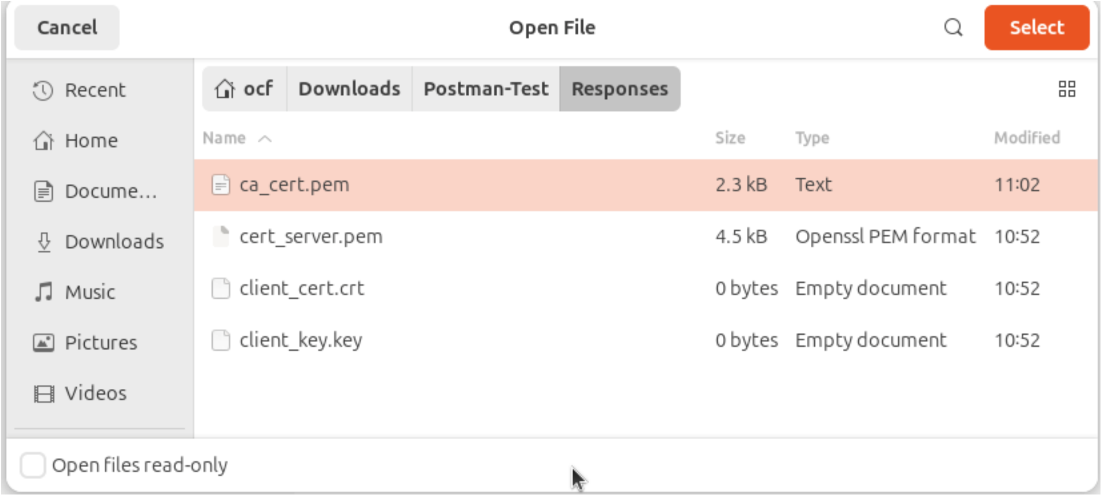
CA certificates

PEM file Select File

The file should consist of one or more trusted certificates in PEM format.

Client certificates

Add and manage SSL certificates on a per domain basis. Learn more about working with certificates ↗

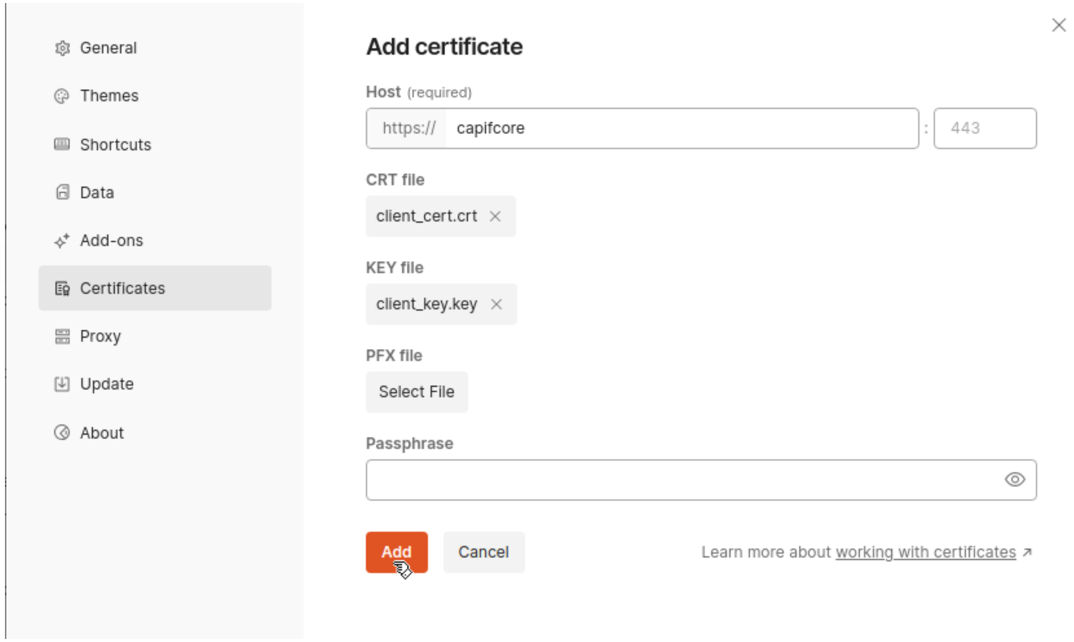


Name	Size	Type	Modified
ca_cert.pem	2.3 kB	Text	11:02
cert_server.pem	4.5 kB	Openssl PEM format	10:52
client_cert.crt	0 bytes	Empty document	10:52
client_key.key	0 bytes	Empty document	10:52

☐ Open files read-only

Configure Certificates in Postman

Now add Client certificates by clicking at “add certificate” button:



Add certificate

Host (required)

https:// capifcore : 443

CRT file

client_cert.crt

KEY file

client_key.key

PFX file

Select File

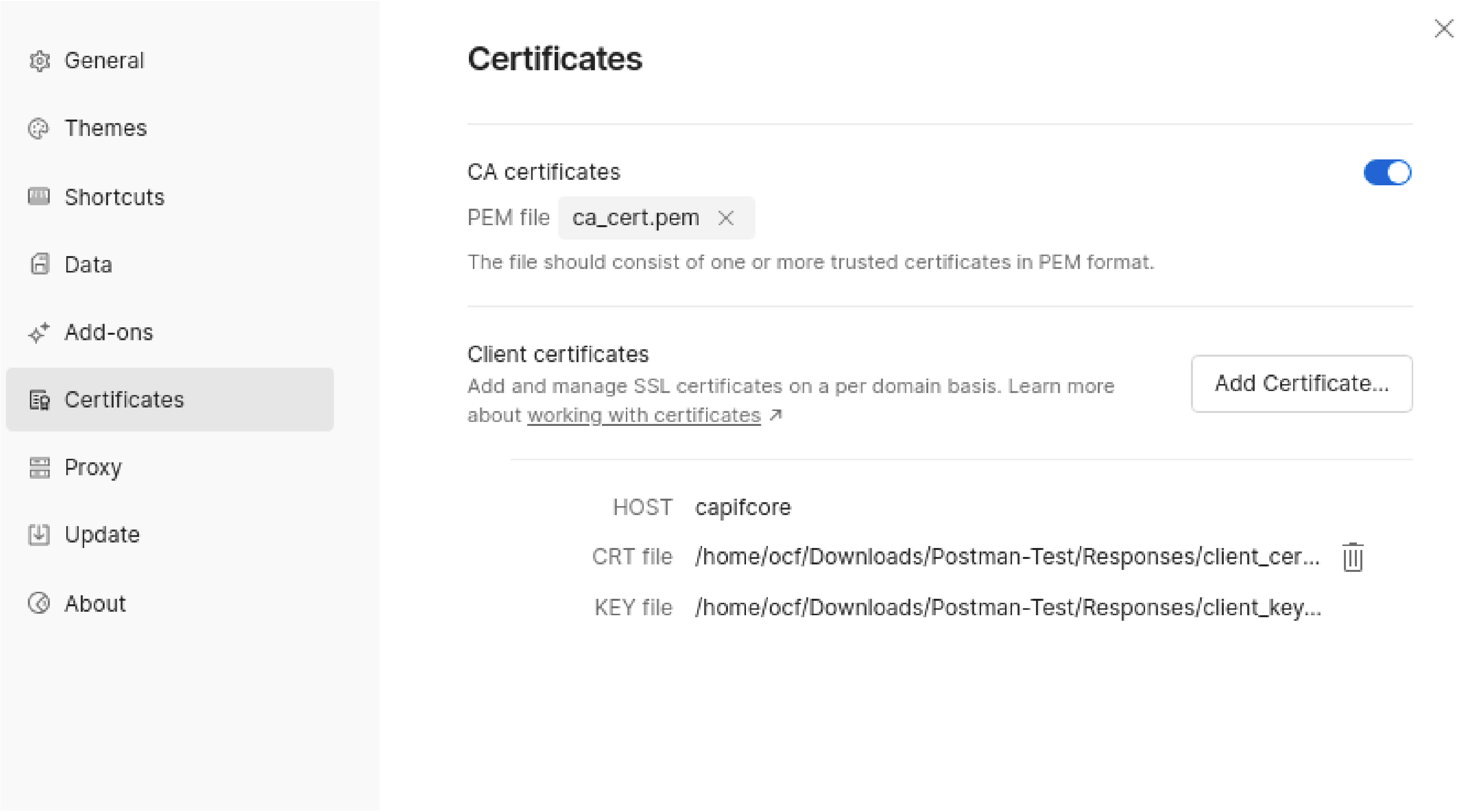
Passphrase

Setup capifcore as host and import client_cert.crt and client_key.key to each field, after that click on “Add”

Configure Certificates in Postman

OpenCAPIF

If all is configured properly you will see this:



Certificates

CA certificates

PEM file ca_cert.pem

The file should consist of one or more trusted certificates in PEM format.

Client certificates

Add and manage SSL certificates on a per domain basis. Learn more about working with certificates ↗

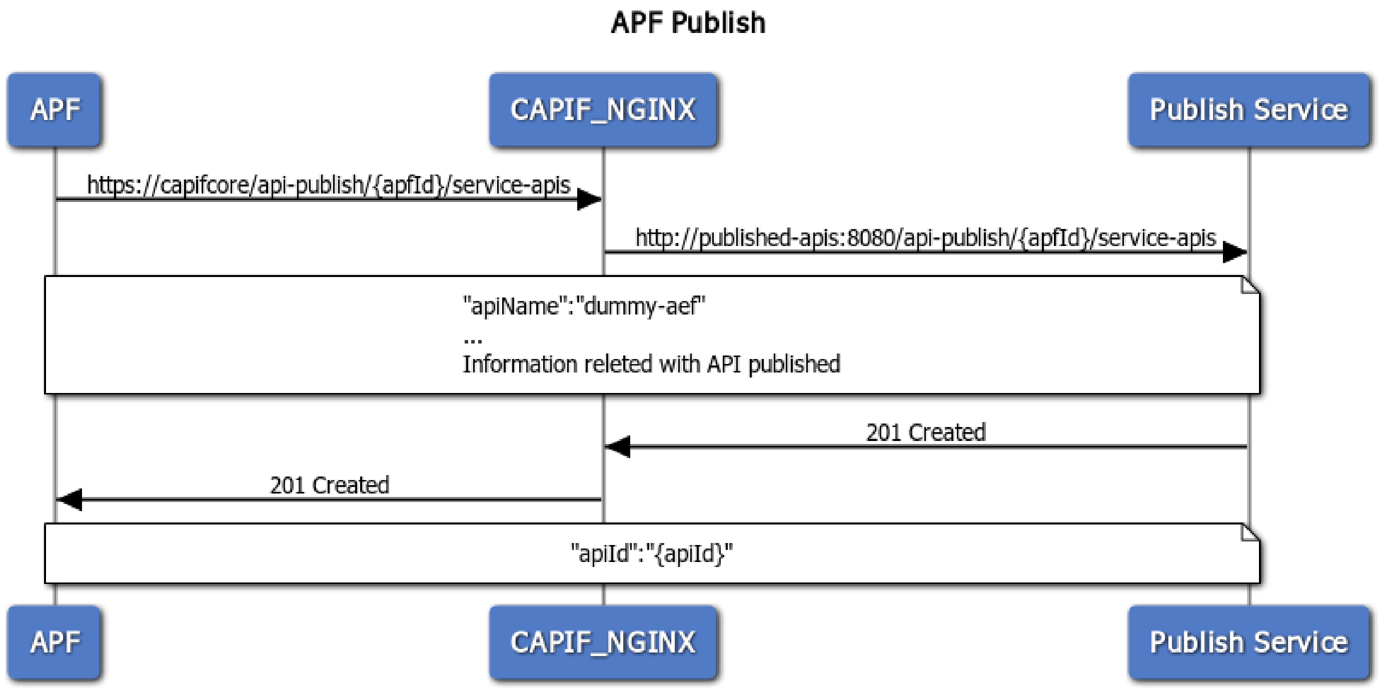
HOST capifcore

CRT file /home/ocf/Downloads/Postman-Test/Responses/client_cer...

KEY file /home/ocf/Downloads/Postman-Test/Responses/client_key...

Publish API by provider

After onboard provider we can publish an API:



APF Publish

APF

CAPIF_NGINX

Publish Service

https://capifcore/api-publish/{apfId}/service-apis

http://published-apis:8080/api-publish/{apfId}/service-apis

"apiName":"dummy-aef"
...
Information releted with API published

201 Created

201 Created

"apiId":"{apiId}"

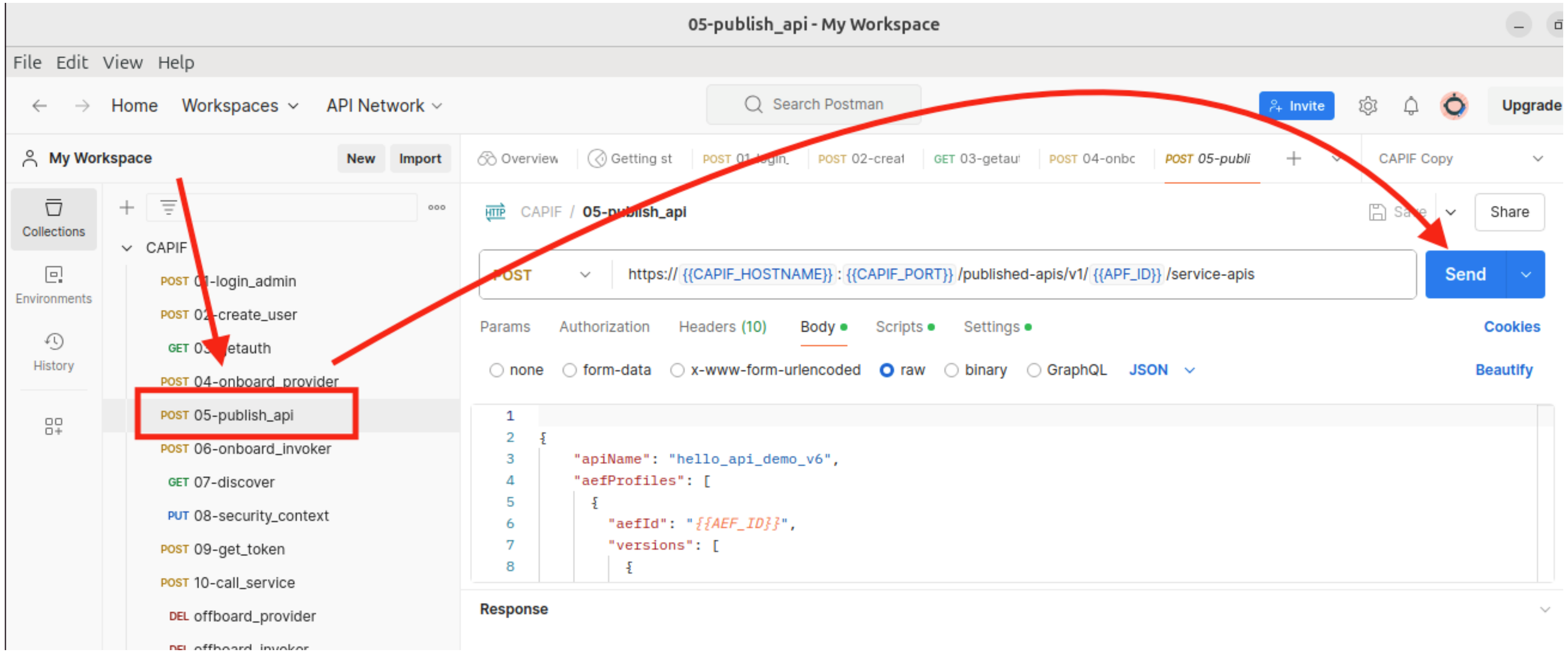
APF

CAPIF_NGINX

Publish Service

Publish API by provider

Now we can select request 05-publish_api:



POST https://{{CAPIF_HOSTNAME}}:{{CAPIF_PORT}}/published-apis/v1/{{APF_ID}}/service-apis

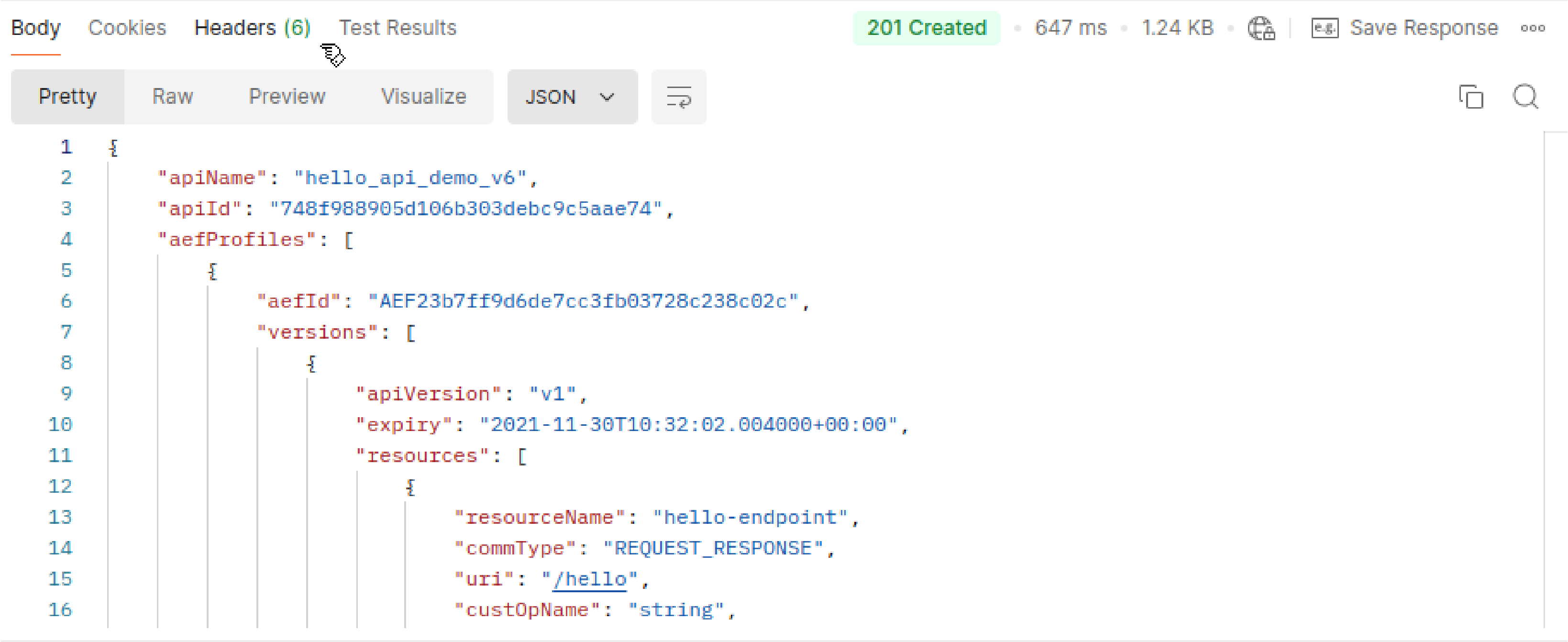
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```

{
    "apiName": "hello_api_demo_v6",
    "aefProfiles": [
      {
        "aefId": "{{AEF_ID}}",
        "versions": [
          {
```

Publish API by provider

Response will be 201 Created with next body:



```
{
    "apiName": "hello_api_demo_v6",
    "apiId": "748f988905d106b303debc9c5aae74",
    "aefProfiles": [
        {
            "aefId": "AEF23b7ff9d6de7cc3fb03728c238c02c",
            "versions": [
                {
                    "apiVersion": "v1",
                    "expiry": "2021-11-30T10:32:02.004000+00:00",
                    "resources": [
                        {
                            "resourceName": "hello-endpoint",
                            "commType": "REQUEST_RESPONSE",
                            "uri": "/hello",
                            "custOpName": "string",
```

You can see here the apiId assigned by CCF to this published API.

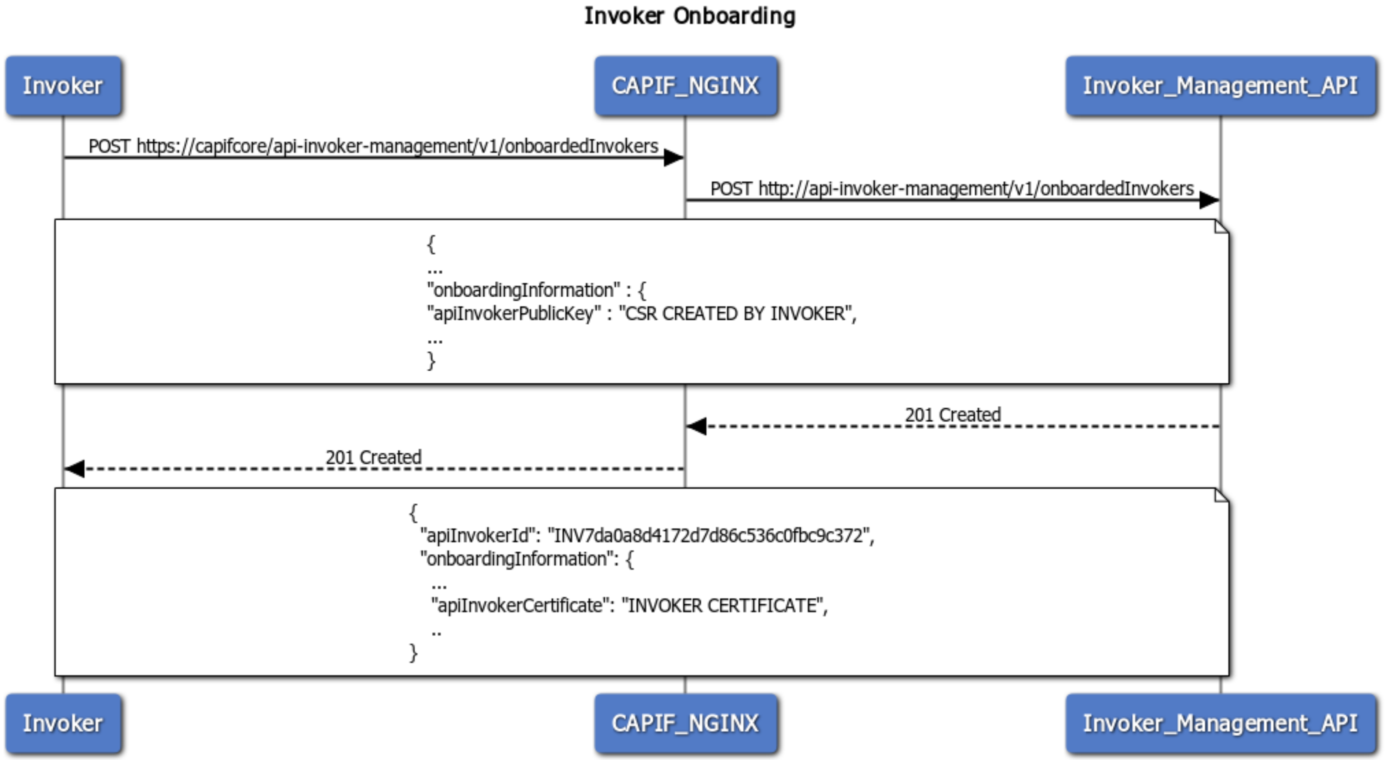
Which are our current status?

- User created by admin
- Provider onboarded
- Provider API published

Invoker Onboarding flow by customer User

Onboard invoker

After API publication we can onboard an invoker on CCF



Invoker Onboarding

POST https://capifcore/api-invoker-management/v1/onboardedInvokers

POST http://api-invoker-management/v1/onboardedInvokers

```
{
...
"onboardingInformation" : {
"apiInvokerPublicKey" : "CSR CREATED BY INVOKER",
...
}
```

201 Created

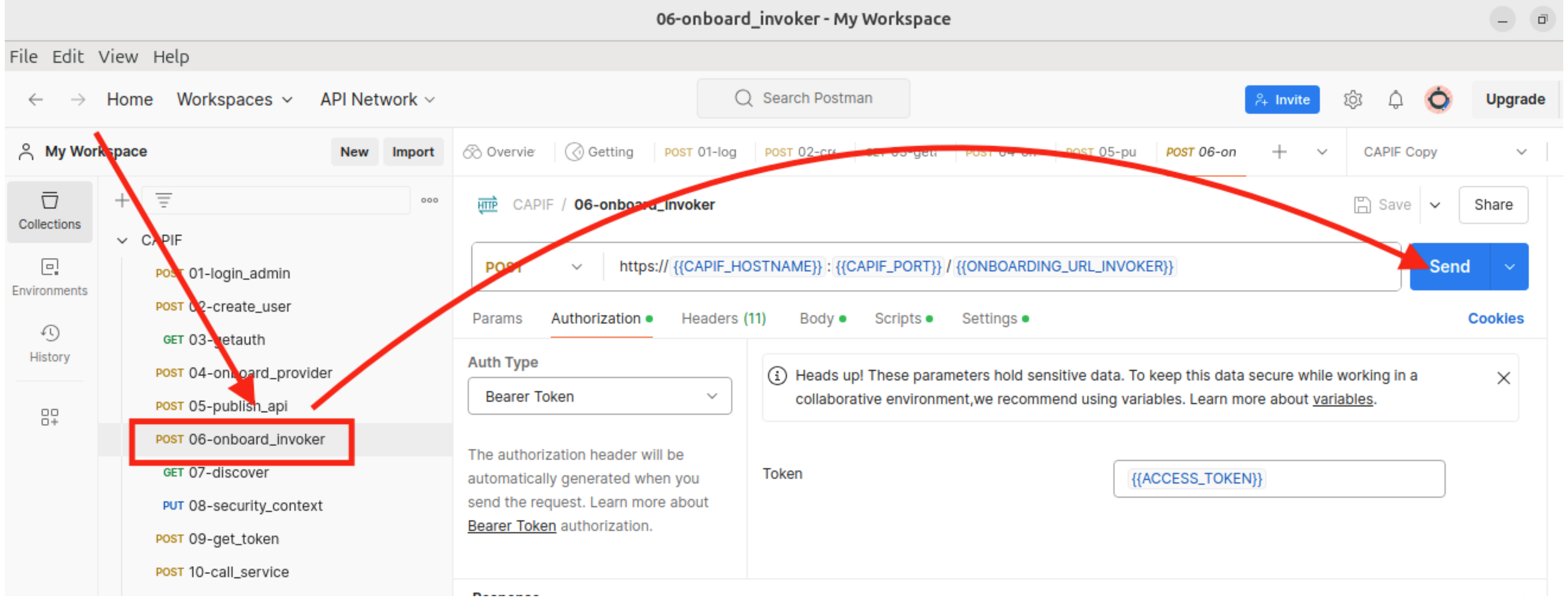
201 Created

```
{
 "apiInvokerId": "INV7da0a8d4172d7d86c536c0fbc9c372",
 "onboardingInformation": {
  ...
  "apiInvokerCertificate": "INVOKER CERTIFICATE",
  ..
}
```

Onboard invoker

OpenCAPIF

Now we will onboard and invoker in CCF. We must select request 06-onboard_invoker and click on “Send”:



CAPIF
- POST 01-login_admin
- POST 02-create_user
- GET 03-getauth
- POST 04-onboard_provider
- POST 05-publish_api
- POST 06-onboard_invoker
- GET 07-discover
- PUT 08-security_context
- POST 09-get_token
- POST 10-call_service

CAPIF / 06-onboard_invoker

POST https:// {{CAPIF_HOSTNAME}} : {{CAPIF_PORT}} / {{ONBOARDING_URL_INVOKER}}

Auth Type Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

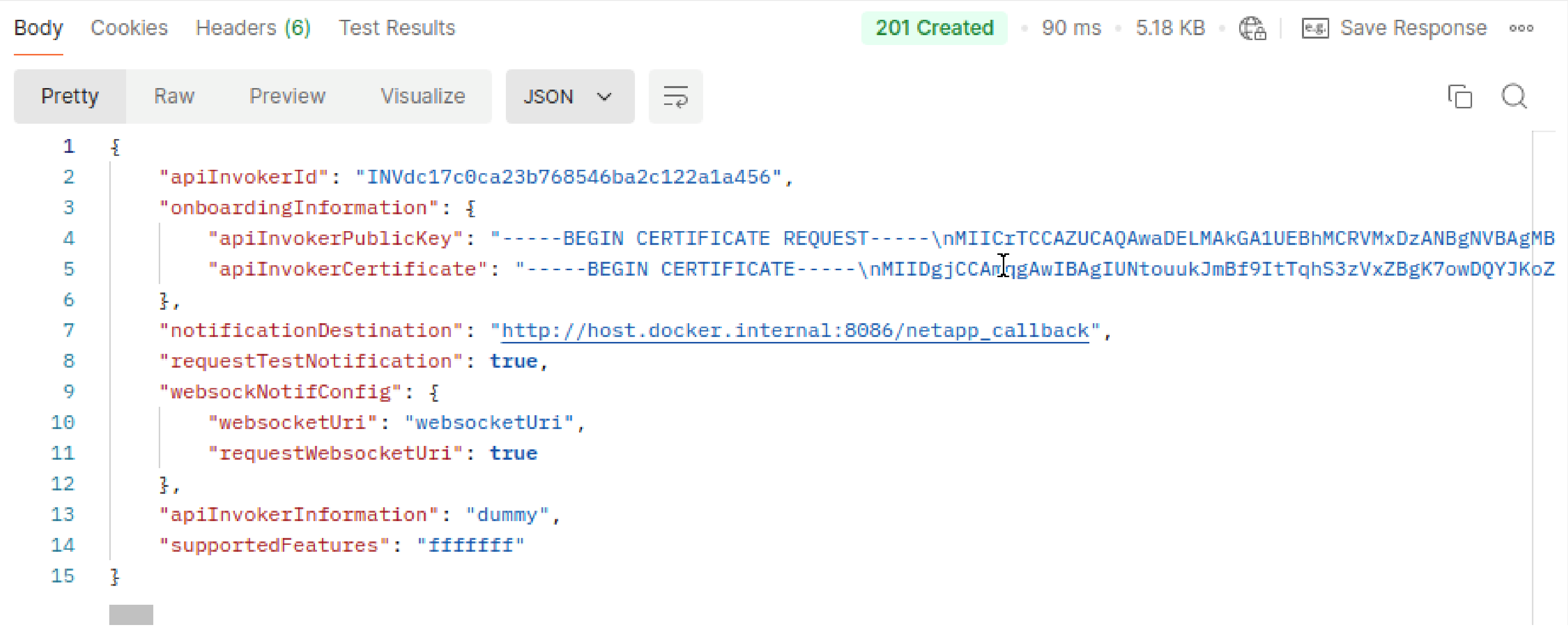
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

Token {{ACCESS_TOKEN}}

As Provider Onboarding, for Invoker onboarding request we must use access token provided by getauth, this is because at onboarding operation we will retrieve the signed certificate to interact with CCF.

Onboard invoker

Response will be 201 Created with signed certificate in body:

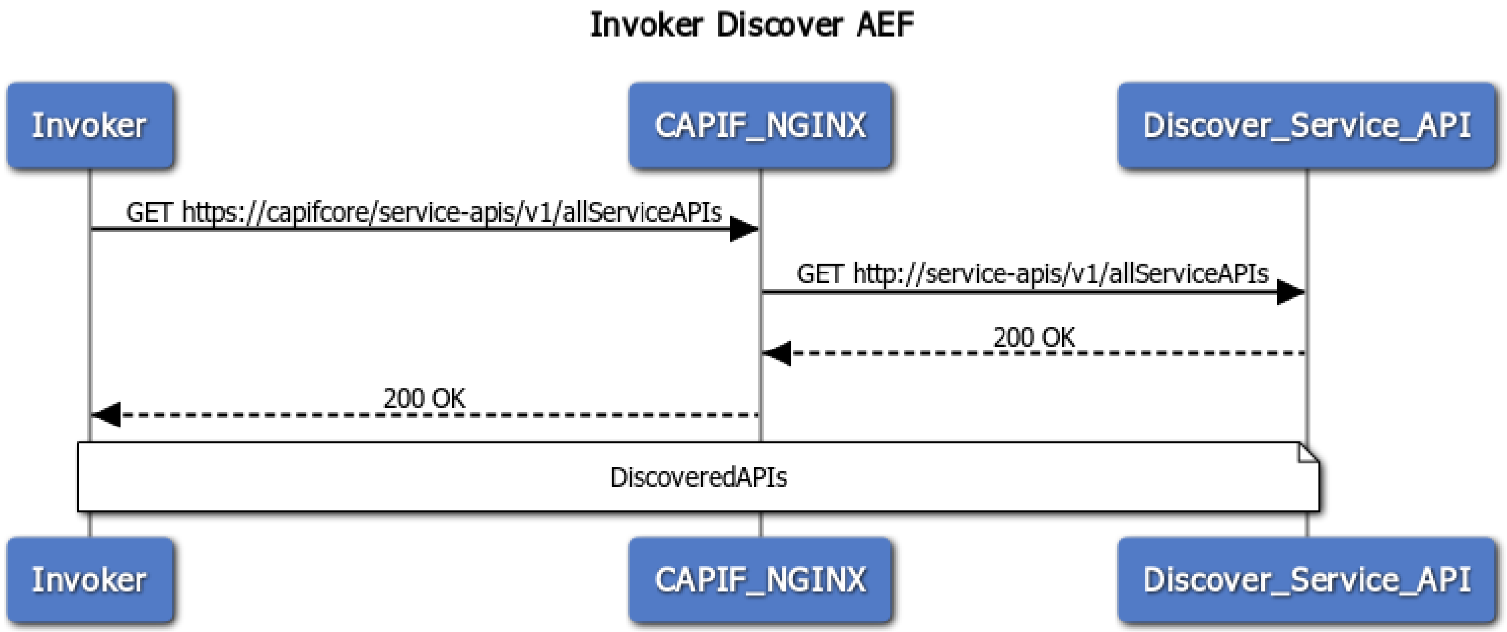


```
{
    "apiInvokerId": "INVdc17c0ca23b768546ba2c122a1a456",
    "onboardingInformation": {
        "apiInvokerPublicKey": "-----BEGIN CERTIFICATE REQUEST-----\nMIICrTCCAZUCAQAwaDELMAkGA1UEBhMCRVMxDzANBgNVBAgMB
        "apiInvokerCertificate": "-----BEGIN CERTIFICATE-----\nMIIDgjCCAmqgAwIBAgIUNtouukJmBf9ItTqhS3zVxZBgK7owDQYJKoZ
    },
    "notificationDestination": "http://host.docker.internal:8086/netapp_callback",
    "requestTestNotification": true,
    "websockNotifConfig": {
        "websocketUri": "websocketUri",
        "requestWebsocketUri": true
    },
    "apiInvokerInformation": "dummy",
    "supportedFeatures": "fffffff"
}
```

Also we can see apiInvokerId provided by CCF to identify this invoker inside CCF.

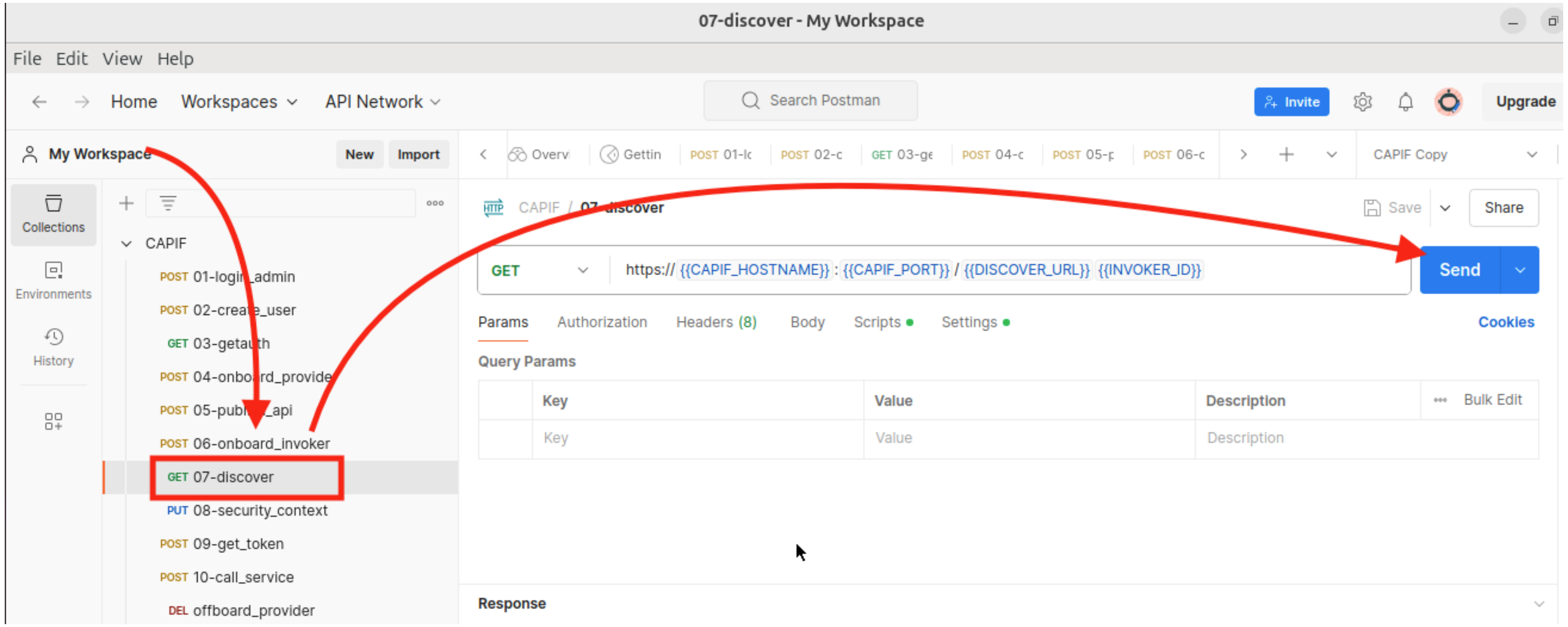
Discover APIs by Invoker

Now we can get all APIs published by request a discover:



Discover APIs by Invoker

We can now select request 07-discover to retrieve APIs published:



07-discover - My Workspace

GET https:// {{CAPIF_HOSTNAME}} : {{CAPIF_PORT}} / {{DISCOVER_URL}} {{INVOKER_ID}}

CAPIF / 07-discover

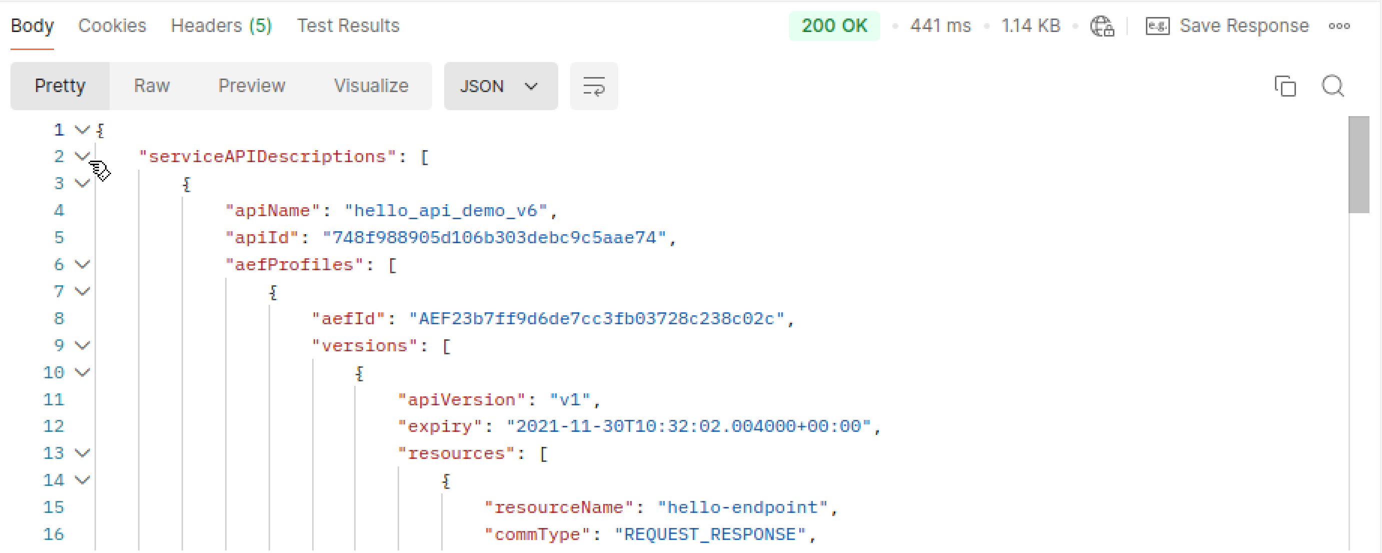
- POST 01-login_admin
- POST 02-create_user
- GET 03-getauth
- POST 04-onboard_provider
- POST 05-publish_api
- POST 06-onboard_invoker
- GET 07-discover
- PUT 08-security_context
- POST 09-get_token
- POST 10-call_service
- DEL offboard_provider

Query Params

Key	Value	Description
Key	Value	Description

APIs discovered

Response will be 200 OK with published APIs:



```
{
    "serviceAPIDescriptions": [
        {
            "apiName": "hello_api_demo_v6",
            "apiId": "748f988905d106b303debc9c5aae74",
            "aefProfiles": [
                {
                    "aefId": "AEF23b7ff9d6de7cc3fb03728c238c02c",
                    "versions": [
                        {
                            "apiVersion": "v1",
                            "expiry": "2021-11-30T10:32:02.004000+00:00",
                            "resources": [
                                {
                                    "resourceName": "hello-endpoint",
                                    "commType": "REQUEST_RESPONSE",
```

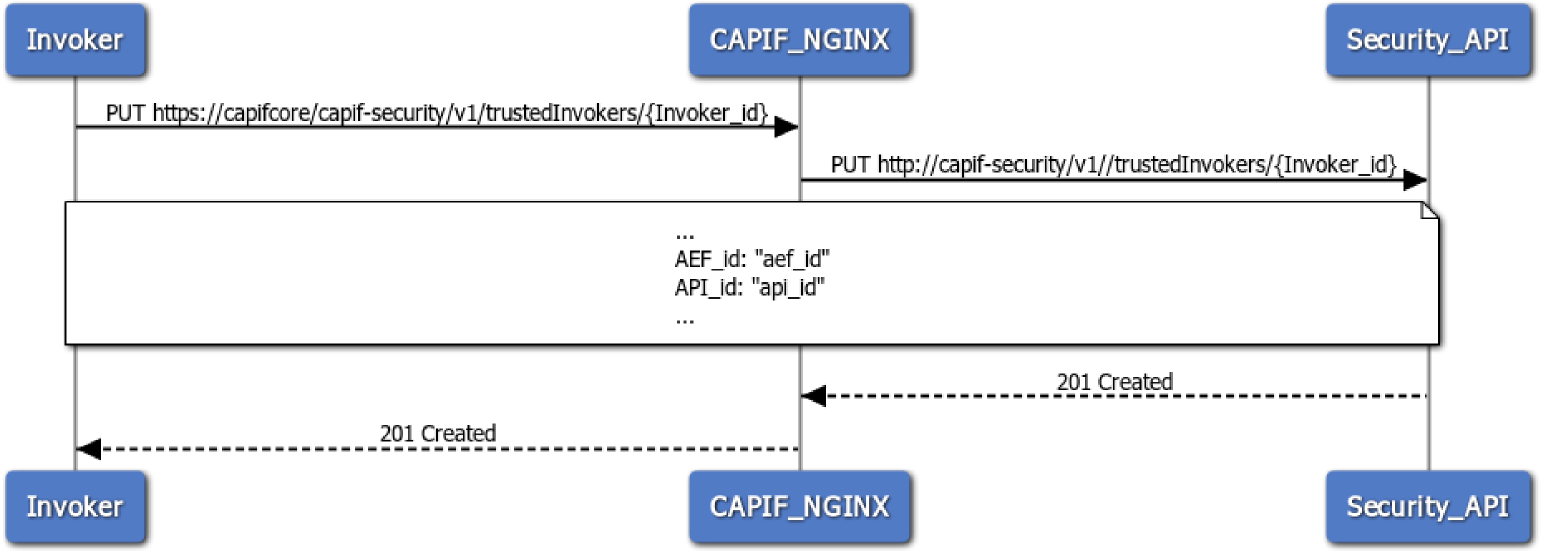
We can see there a list of serviceAPIDescriptions with all APIs published at CCF.

We will try to use hello_api_demo_v6 api.

Create Security Context for that API

If invoker want to use some discovered API, then security context must be requested:

Invoker Create Security Context



PUT https://capifcore/capif-security/v1/trustedInvokers/{Invoker_id}

PUT http://capif-security/v1//trustedInvokers/{Invoker_id}

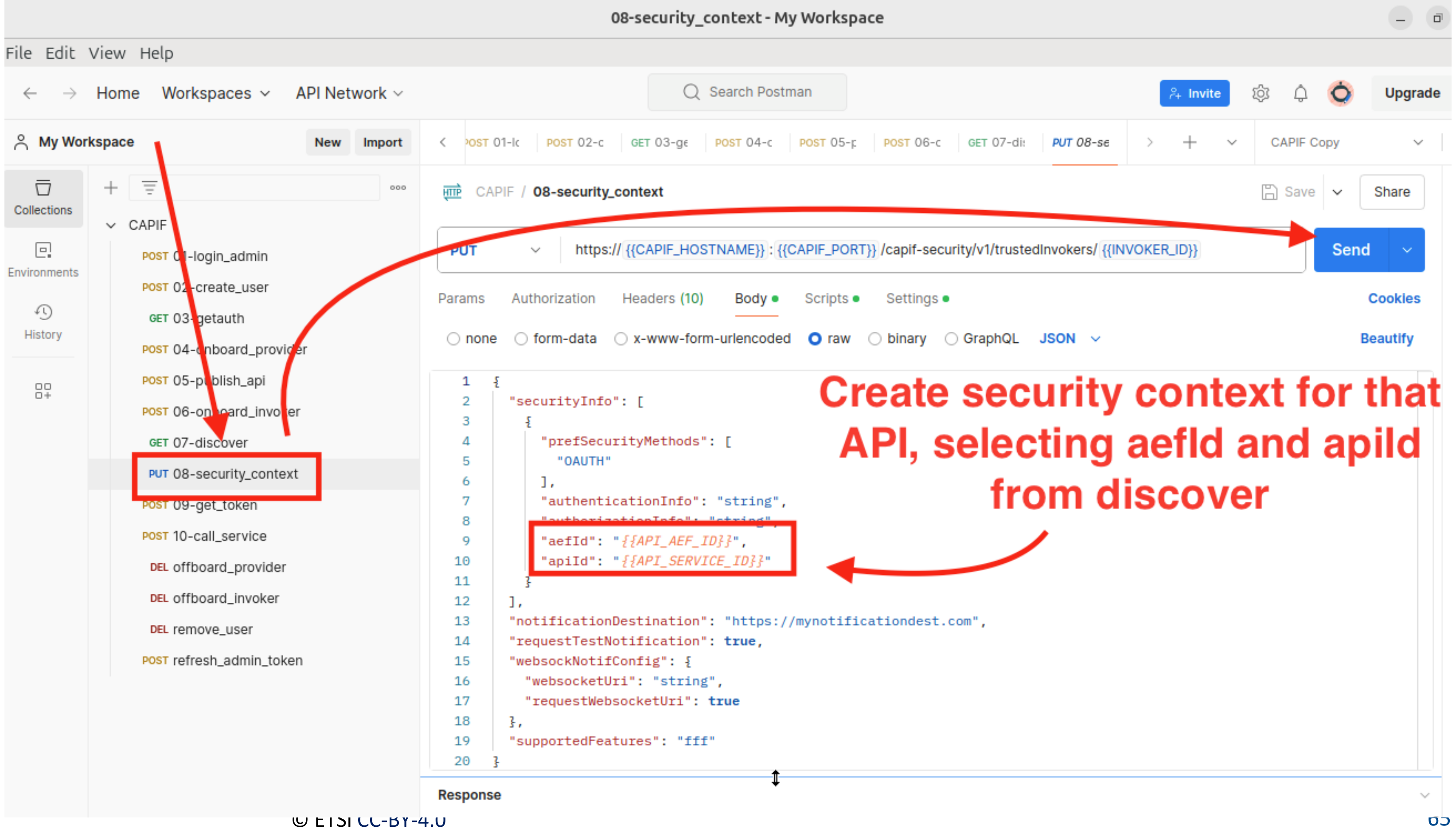
```
...
AEF_id: "aef_id"
API_id: "api_id"
...
```

201 Created

201 Created

Create Security Context for that API

We need to request a Security Context as invoker to grant access to selected API:



PUT https:// {{CAPIF_HOSTNAME}} : {{CAPIF_PORT}} /capif-security/v1/trustedInvokers/ {{INVOKER_ID}}

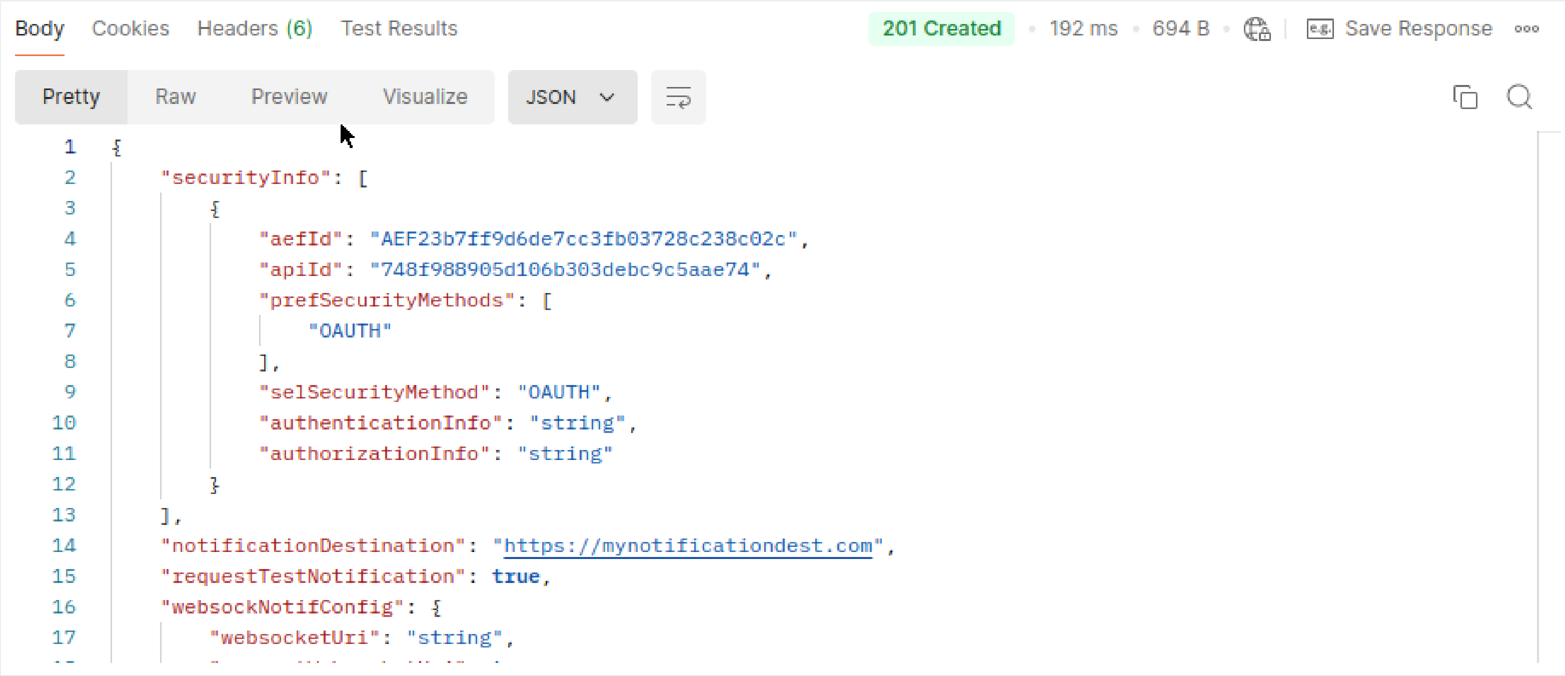
Body: ○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
  "securityInfo": [
    {
      "prefSecurityMethods": [
        "OAUTH"
      ],
      "authenticationInfo": "string",
      "authorizationInfo": "string",
      "aefId": "{{API_AEF_ID}}",
      "apiId": "{{API_SERVICE_ID}}"
    }
  ],
  "notificationDestination": "https://mynotificationdest.com",
  "requestTestNotification": true,
  "websockNotifConfig": {
    "websocketUri": "string",
    "requestWebsocketUri": true
  },
  "supportedFeatures": "fff"
}
```

Create security context for that API, selecting aefId and apiId from discover

Security Context created

Response will be 201 Created with Security Info:



```
{
    "securityInfo": [
        {
            "aefId": "AEF23b7ff9d6de7cc3fb03728c238c02c",
            "apiId": "748f988905d106b303debc9c5aae74",
            "prefSecurityMethods": [
                "OAUTH"
            ],
            "selSecurityMethod": "OAUTH",
            "authenticationInfo": "string",
            "authorizationInfo": "string"
        }
    ],
    "notificationDestination": "https://mynotificationdest.com",
    "requestTestNotification": true,
    "websockNotifConfig": {
        "websocketUri": "string",
```

On Release 1 only support OAUTH, then the next step will be get OAUTH token to be used by invoker to access API published.

OpenCAPIF

Which are our current status?

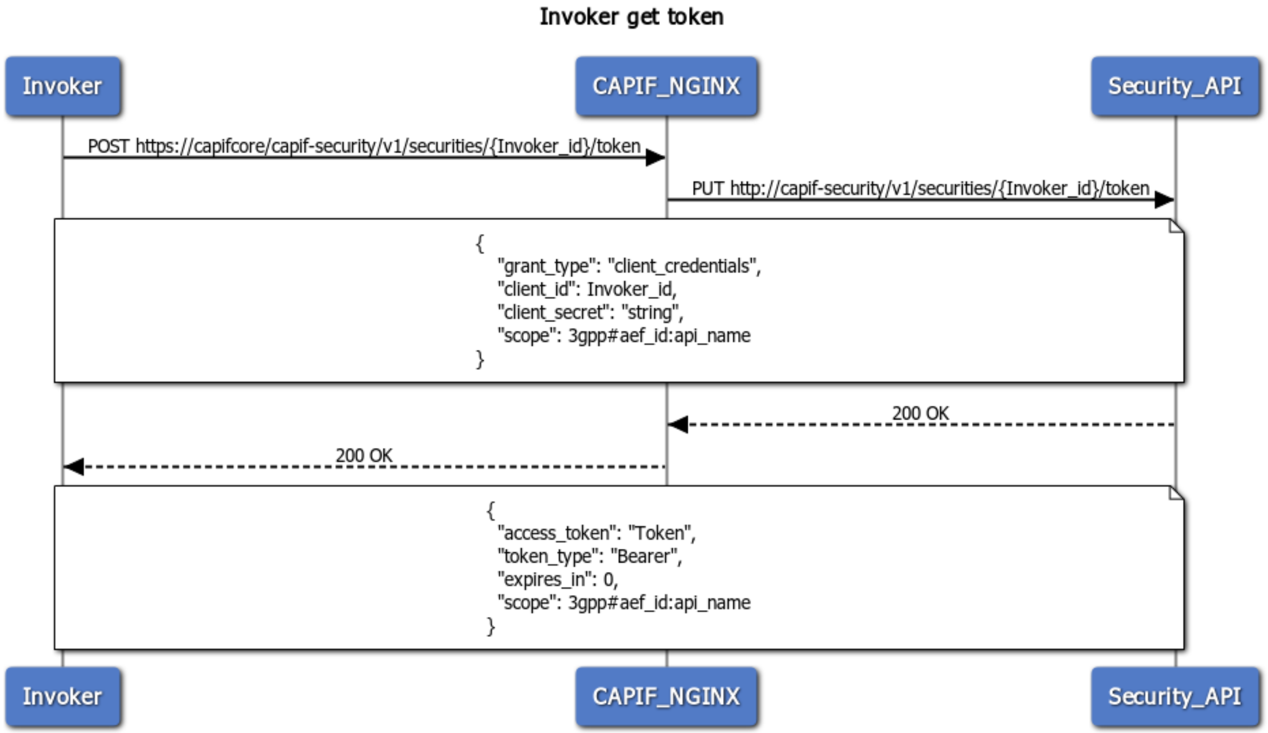
- ◉ Invoker Onboarded
- ◉ APIs Discovered
- ◉ Security Context created

We are ready to reach service API published!

Get OAUTH token

Now the last step will be request OAUTH token to access published service API:

Invoker get token

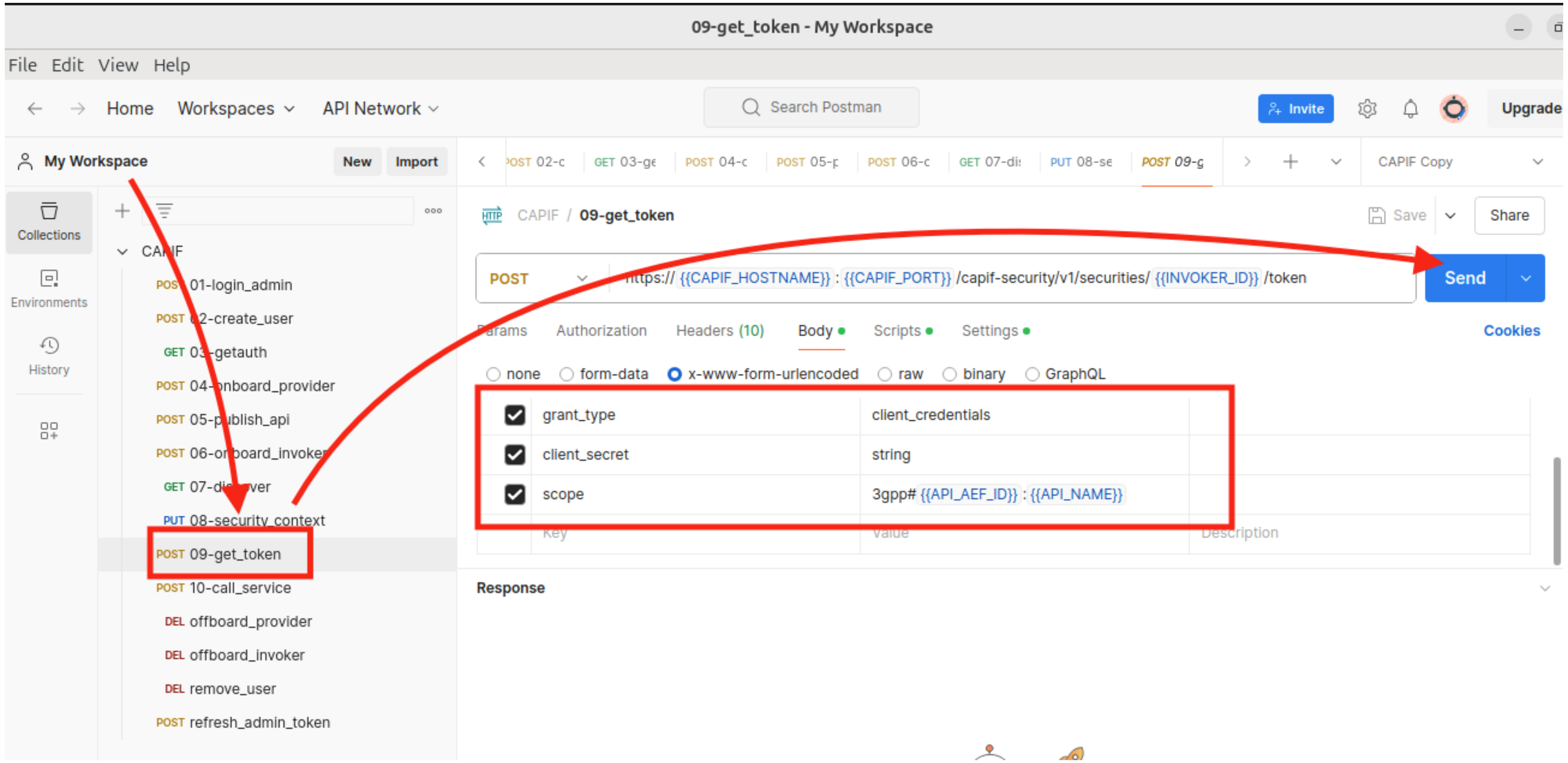

Invoker
CAPIF_NGINX
Security_API
POST https://capifcore/capif-security/v1/securities/{Invoker_id}/token
PUT http://capif-security/v1/securities/{Invoker_id}/token
200 OK
200 OK
Invoker
CAPIF_NGINX
Security_API

```
{
    "grant_type": "client_credentials",
    "client_id": Invoker_id,
    "client_secret": "string",
    "scope": 3gpp#aef_id:api_name
}
```

```
{
    "access_token": "Token",
    "token_type": "Bearer",
    "expires_in": 0,
    "scope": 3gpp#aef_id:api_name
}
```

Get OAUTH token

Select request 09-get_token and click on “Send”:



09-get_token - My Workspace

File Edit View Help

Home Workspaces API Network Search Postman Invite Upgrade

My Workspace New Import

Collections Environments History

CAPIF
- POST 01-login_admin
- POST 02-create_user
- GET 03-getauth
- POST 04-onboard_provider
- POST 05-publish_api
- POST 06-onboard_invoker
- GET 07-discover
- PUT 08-security_context
- POST 09-get_token
- POST 10-call_service
- DEL offboard_provider
- DEL offboard_invoker
- DEL remove_user
- POST refresh_admin_token

CAPIF / 09-get_token Save Share

POST https:// {{CAPIF_HOSTNAME}} : {{CAPIF_PORT}} /capif-security/v1/securities/ {{INVOKER_ID}} /token Send

Params Authorization Headers (10) Body ● Scripts ● Settings ● Cookies

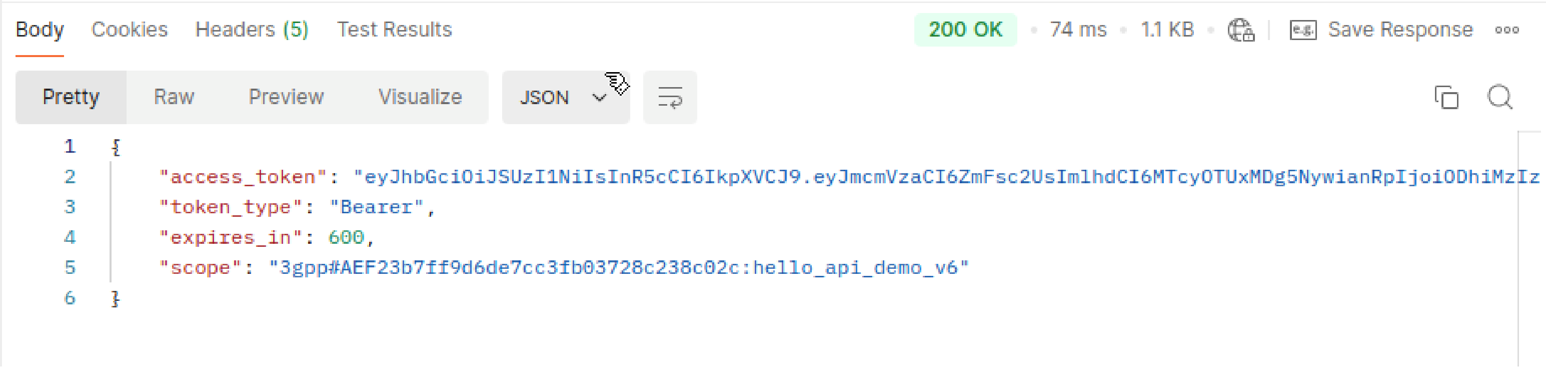
○ none ○ form-data ● x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	Key	Value	Description
☑	grant_type	client_credentials	
☑	client_secret	string	
☑	scope	3gpp#{{API_AEF_ID}}:{{API_NAME}}	

Response

Get OAUTH token

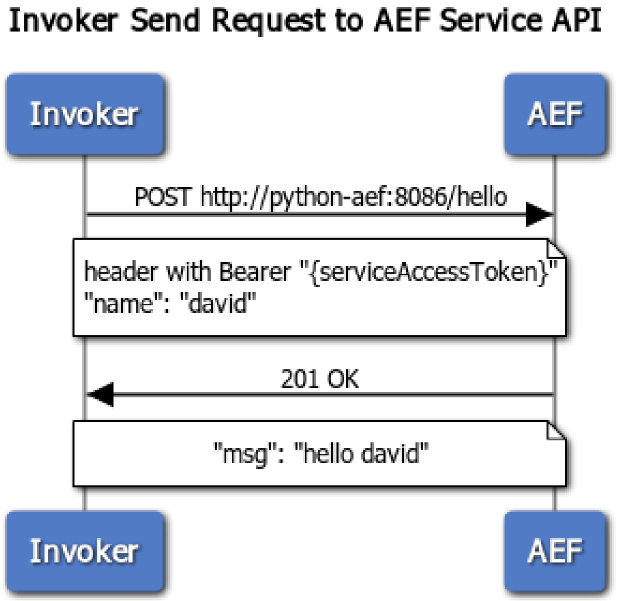
We will receive 200 OK with access_token to be used by Invoker:



```
{
    "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2UsImlhdCI6MTcyOTUxMDg5NywianRpIjoiODhiMzIz
    "token_type": "Bearer",
    "expires_in": 600,
    "scope": "3gpp#AEF23b7ff9d6de7cc3fb03728c238c02c:hello_api_demo_v6"
}
```

Send Request to API published by Provider

Last step is send request using OAUTH to service API published:



Invoker Send Request to AEF Service API

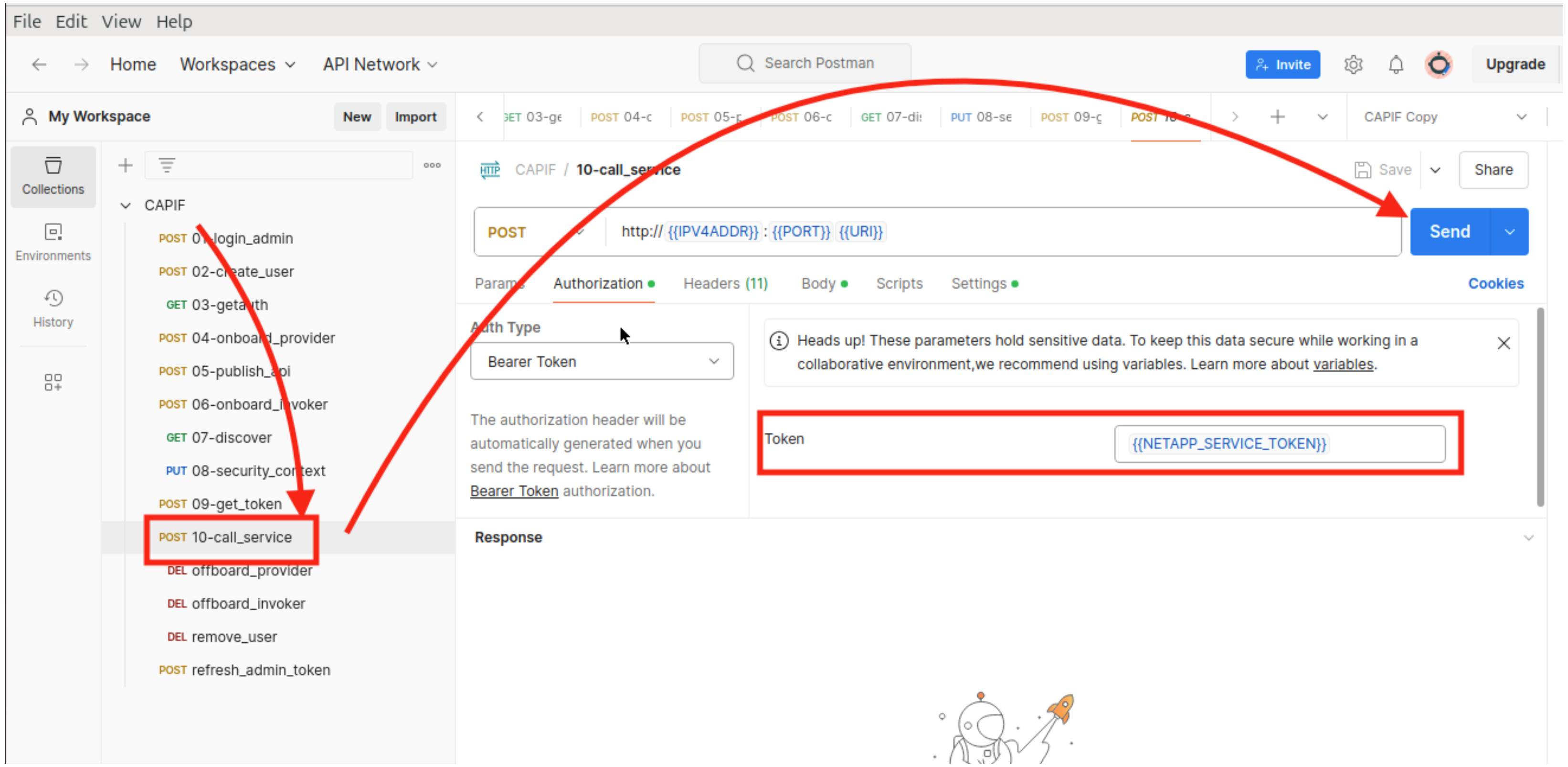
header with Bearer "{serviceAccessToken}"
"name": "david"

"msg": "hello david"

Send Request to API published by Provider

OpenCAPIF

Select request 10-call_service and click on “Send”:



http:// {{IPV4ADDR}} : {{PORT}} {{URI}}

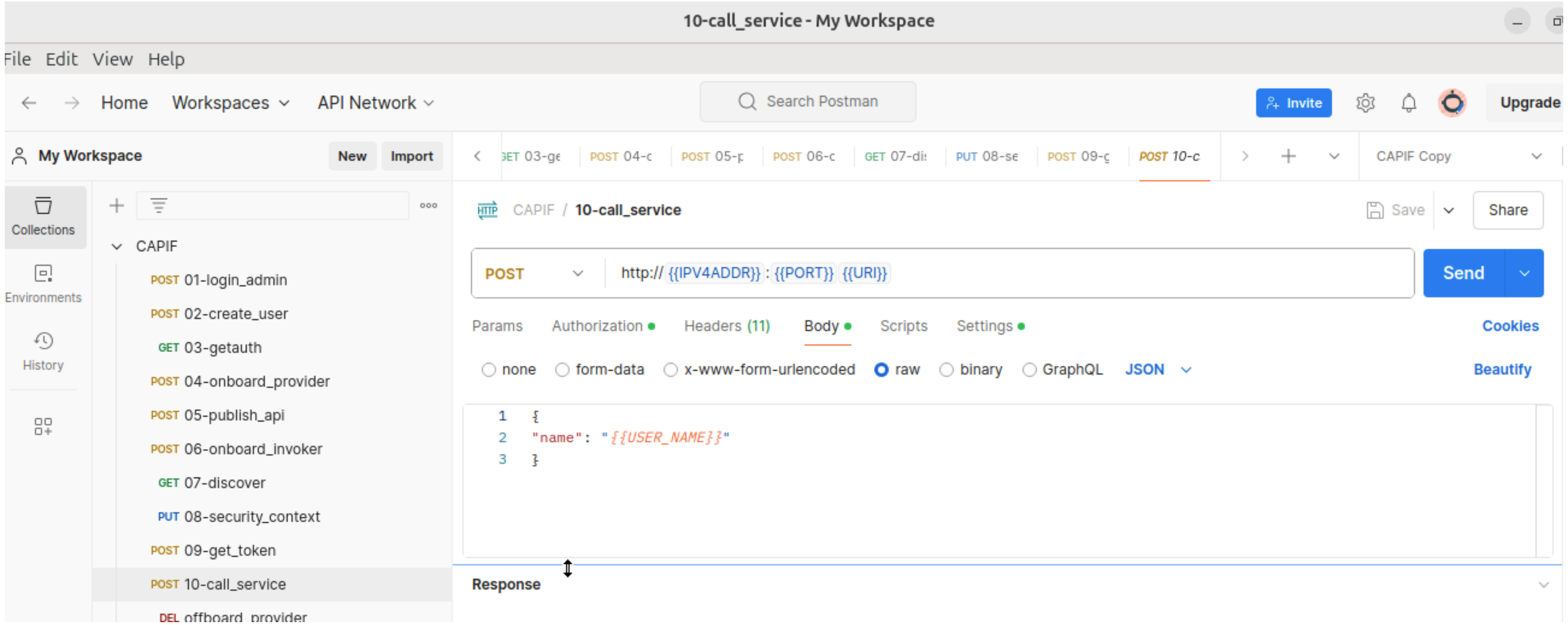
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

Token {{NETAPP_SERVICE_TOKEN}}

Send Request to API published by Provider

Body in request is one defined by API published:



10-call_service - My Workspace

CAPIF / 10-call_service

POST http:// {{IPV4ADDR}} : {{PORT}} {{URI}}

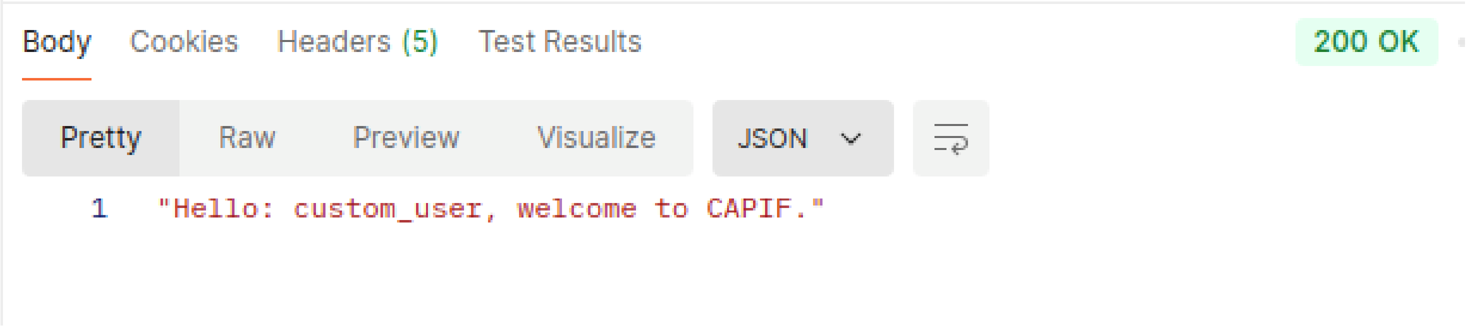
Body: ○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
"name": "{{USER_NAME}}"
}
```

Response from API

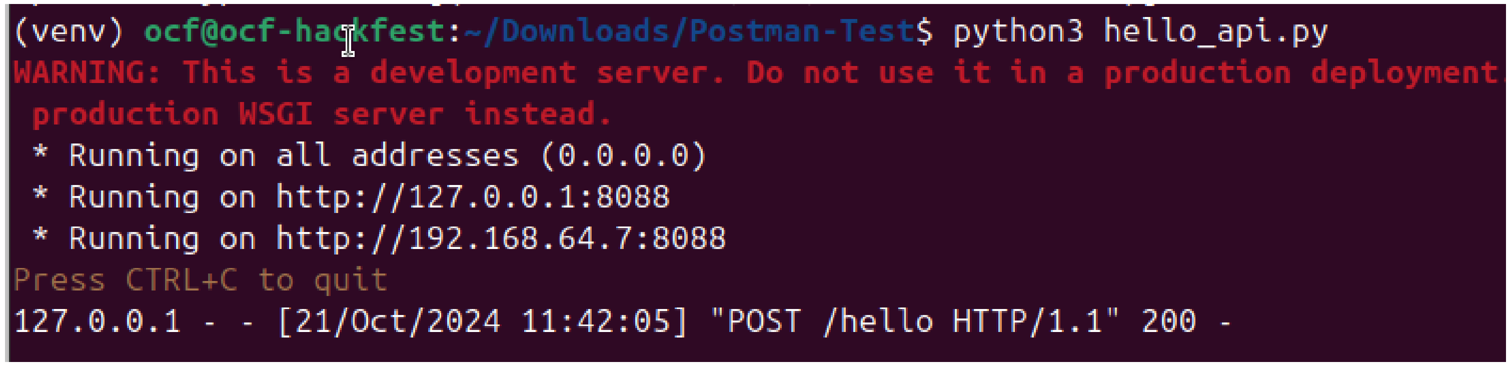
OpenCAPIF

The response will be 200 OK



```
"Hello: custom_user, welcome to CAPIF."
```

We can check the logs of service that is running the API published in terminal:



```
(venv) ocf@ocf-hackfest:~/Downloads/Postman-Test$ python3 hello_api.py
WARNING: This is a development server. Do not use it in a production deployment.
 production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8088
 * Running on http://192.168.64.7:8088
Press CTRL+C to quit
127.0.0.1 - - [21/Oct/2024 11:42:05] "POST /hello HTTP/1.1" 200 -
```

OpenCAPIF

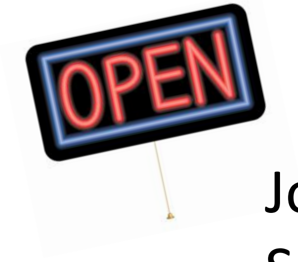
Congratulations!

Now, all of you have completed a full flow using a local deployment of OpenCAPIF! 🎉

Thanks for your attention, everyone! Let’s keep up the great work! 🙌

Engage with OpenCAPIF

Participation is free for ETSI members, SMEs, Universities, Public Research Bodies and User and Trade Associations and Individuals.



Join us by signing the
SDG OCF Agreement

https://portal.etsi.org/ocf

https://ocf.etsi.org

https://labs.etsi.org/rep/ocf

@OpenCAPIF

@OpenCAPIF

@OpenCAPIF

https://OpenCAPIF.slack.com (invite)

OCF_INFO@list.etsi.org

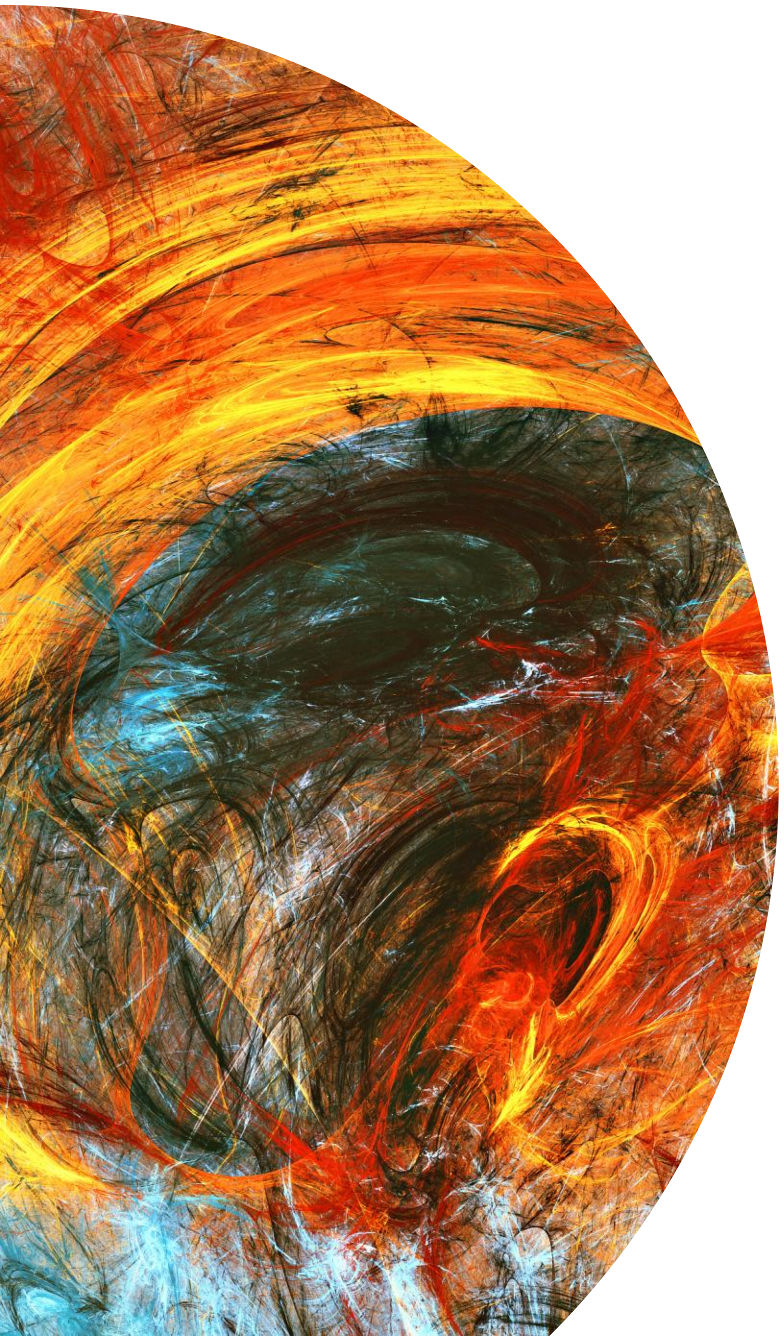
OpenCAPIF

by ETSI

Thank You!

Extras

Overview of OpenCAPIF



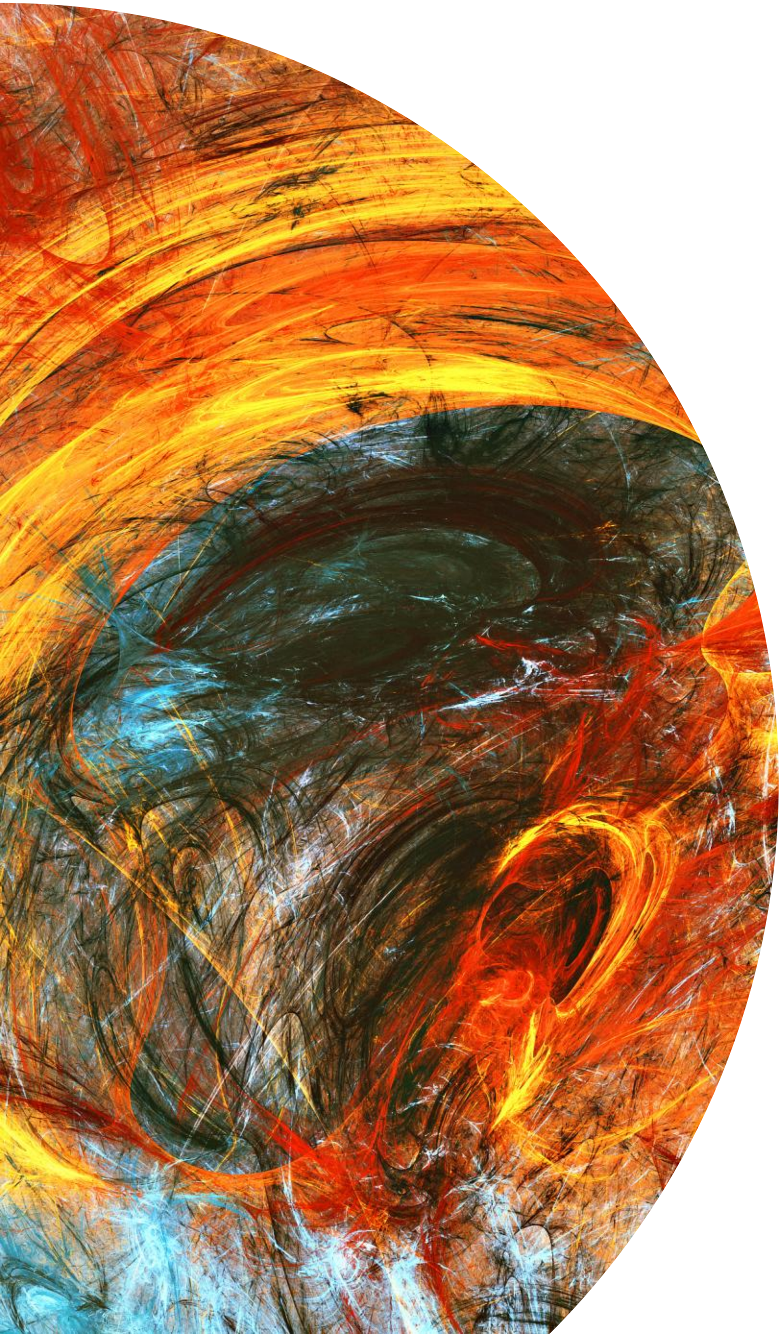
How is OpenCAPIF Created?

OpenCAPIF implements the 3GPP Common API Framework defined on their specs.

The template with models and operations is created by using OpenAPI generator with swaggers created by 3GPP. This simplify the way to update to new releases over 3GPP specifications.

The code logic implemented is under core folder on each service, in order to make easy the update previously commented.

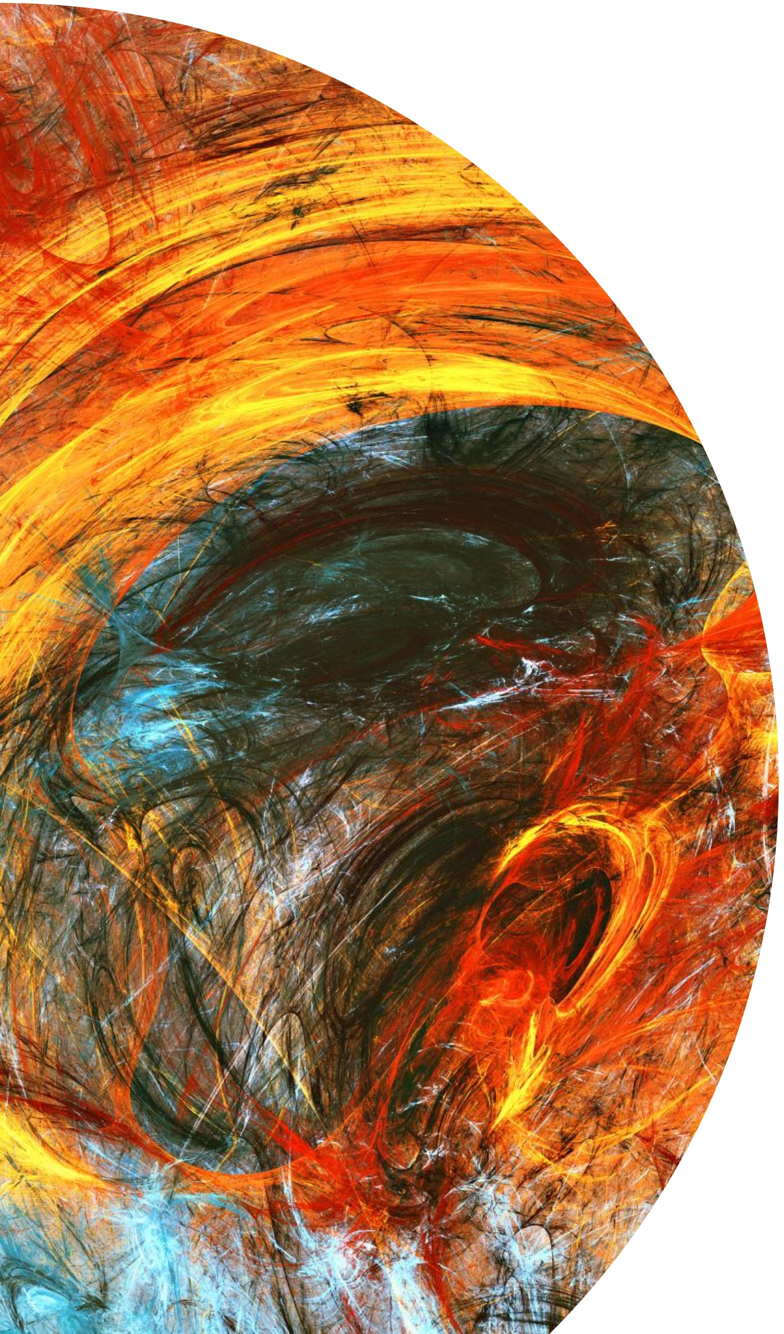
Each API is implemented in a dockerized service, this simplify the way to deploy locally or in a k8s environment.



How is OpenCAPIF Created?

OpenCAPIF use next additional software and libraries:

- Docker
- REDIS
- Python Flask.
- MongoDB
- Mongo Express
- NGINX
- Vault

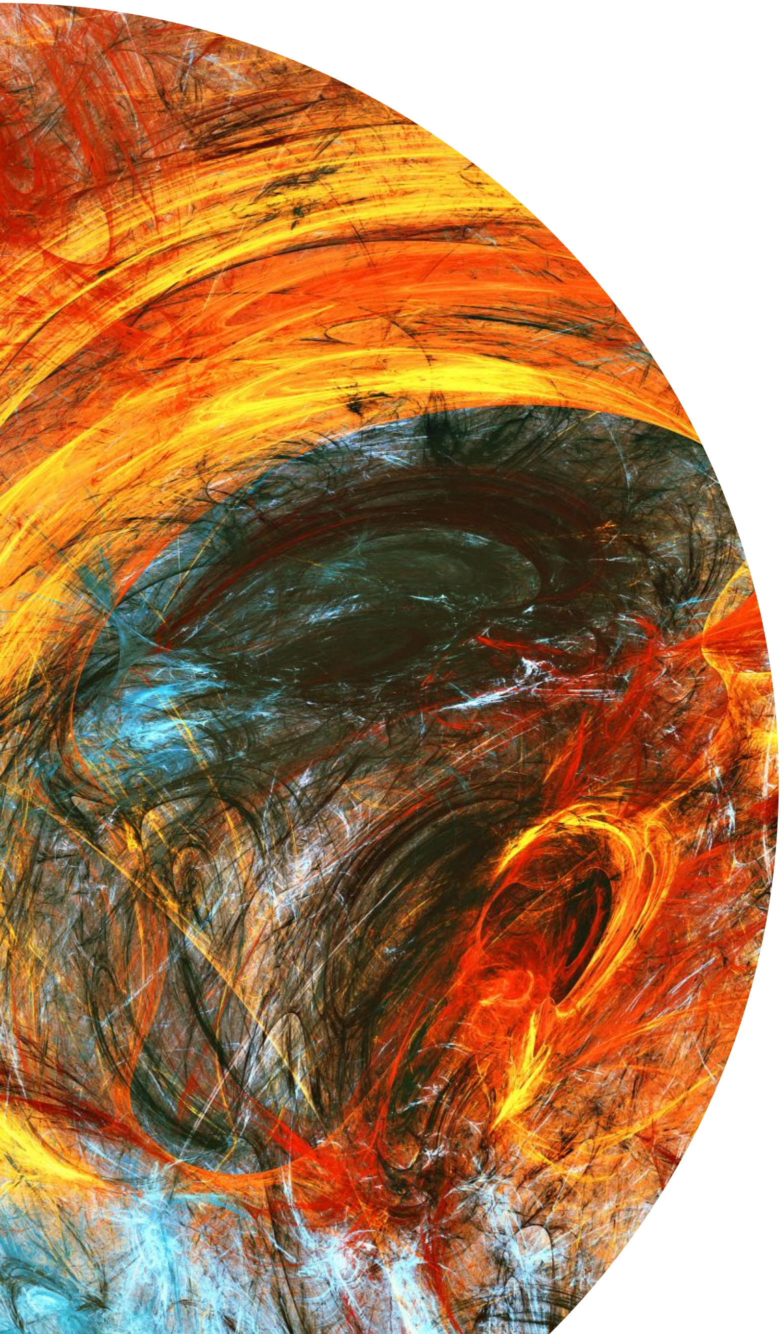


How is OpenCAPIF Created?

On the other hand, at repository we also have a Helm section, which includes:

- Helm Charts:
  - OpenCAPIF microservices.
  - Register Service
  - Vault service
  - Monitoring services.
- Scripts to simplify:
  - Way to deploy all services.
  - Testing over deployed service.

New SDK



New SDK

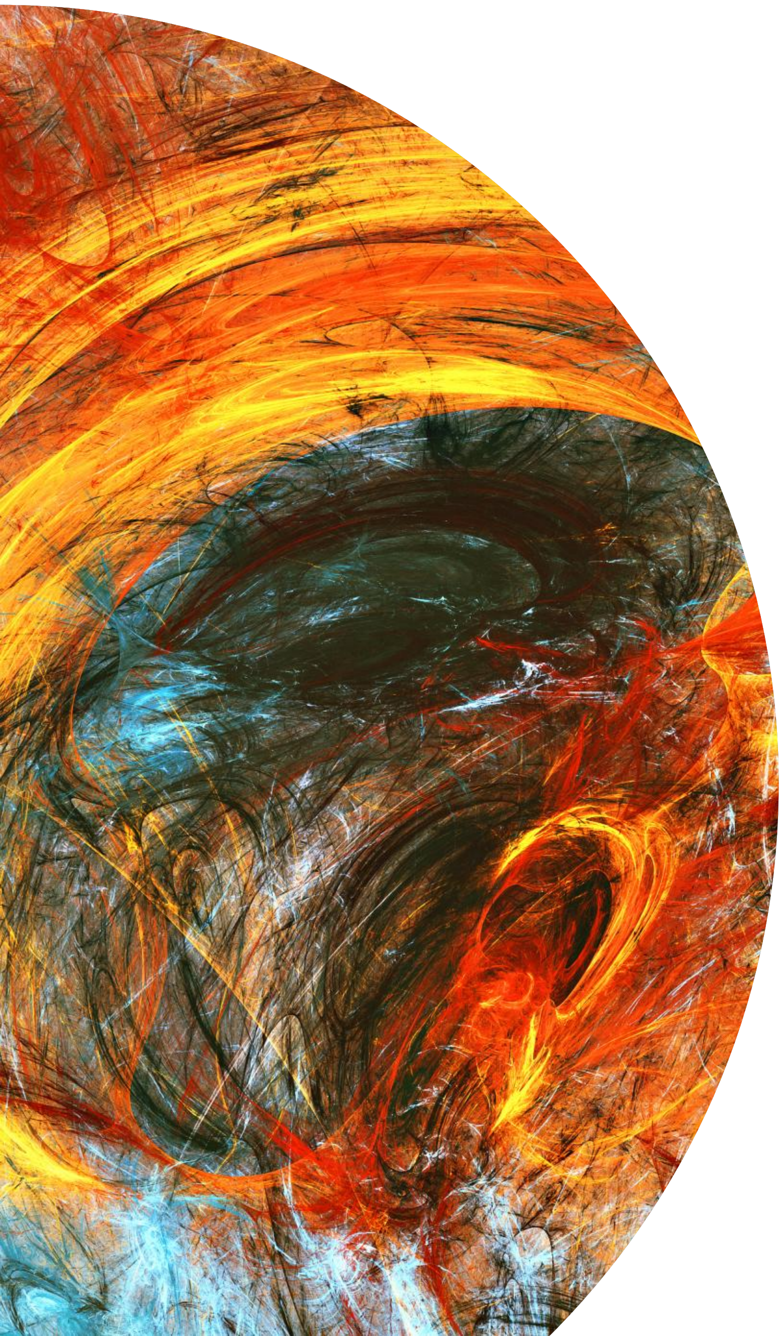
We are working on SDK to be released together with OpenCAPIF Release 2 at January.

This OpenCAPIF SDK brings a set of functions to integrate with the 5G Core's function CAPIF, as defined in 3GPP.

The OpenCAPIF SDK is created as python library, and it will be public to be installed by pip. It will simplify the way to create invokers and providers and their interaction with any OpenCAPIF deployed.

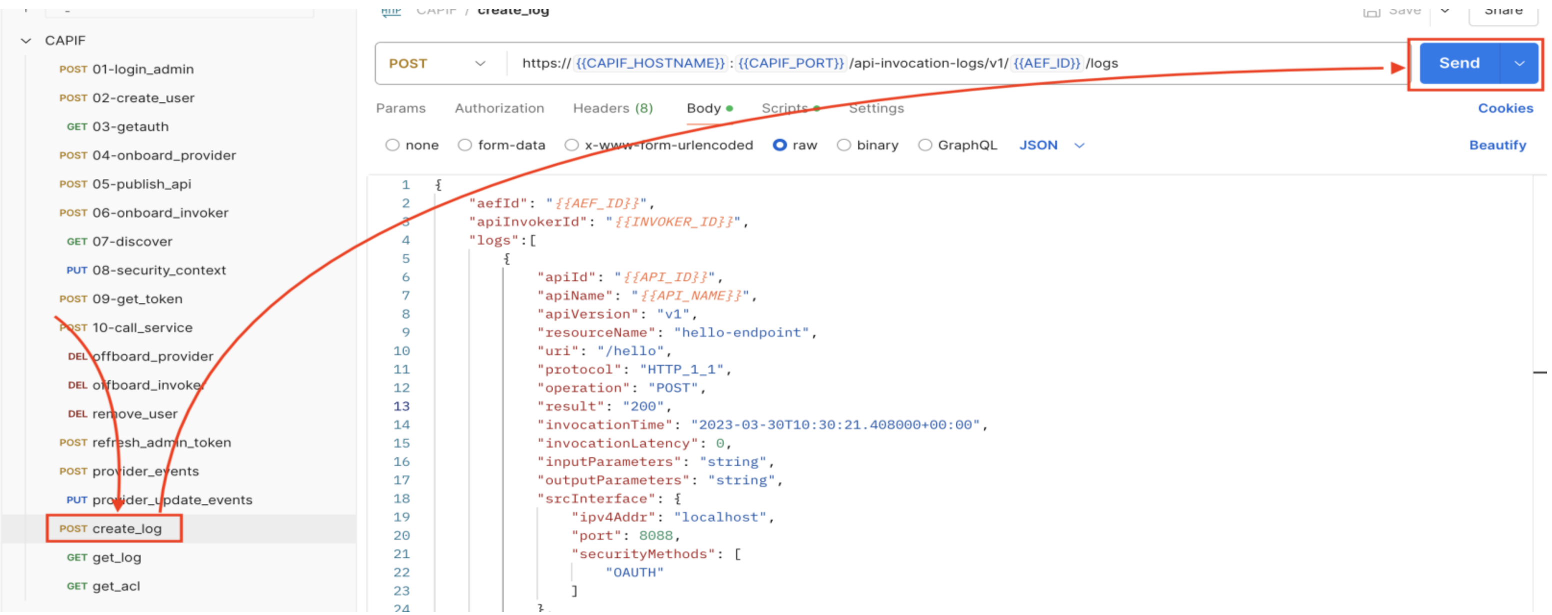
OpenCAPIF

New Postman Requests



Create a Log by provider

Now we can select request create_log:



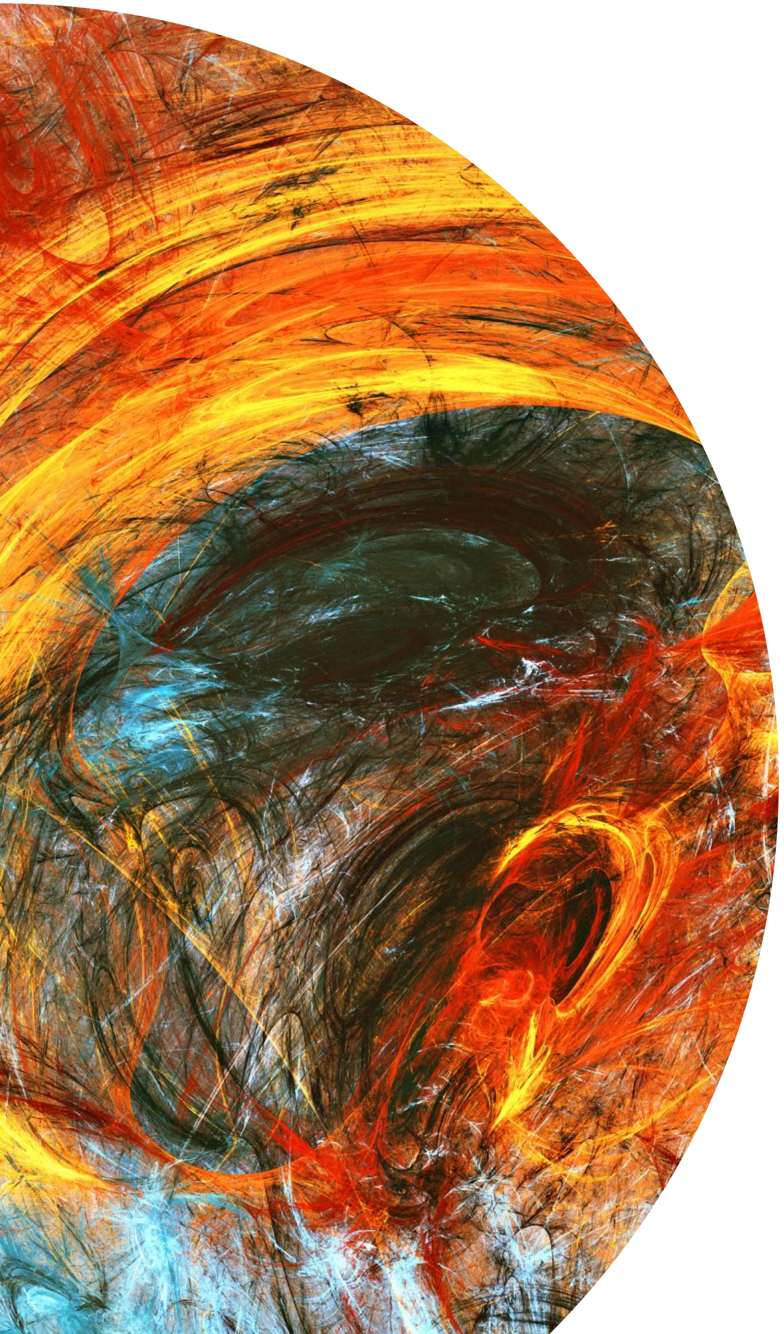
CAPIF

- POST 01-login_admin
- POST 02-create_user
- GET 03-getauth
- POST 04-onboard_provider
- POST 05-publish_api
- POST 06-onboard_invoker
- GET 07-discover
- PUT 08-security_context
- POST 09-get_token
- POST 10-call_service
- DEL offboard_provider
- DEL offboard_invoker
- DEL remove_user
- POST refresh_admin_token
- POST provider_events
- PUT provider_update_events
- POST create_log
- GET get_log
- GET get_acl

CAPIF / create_log

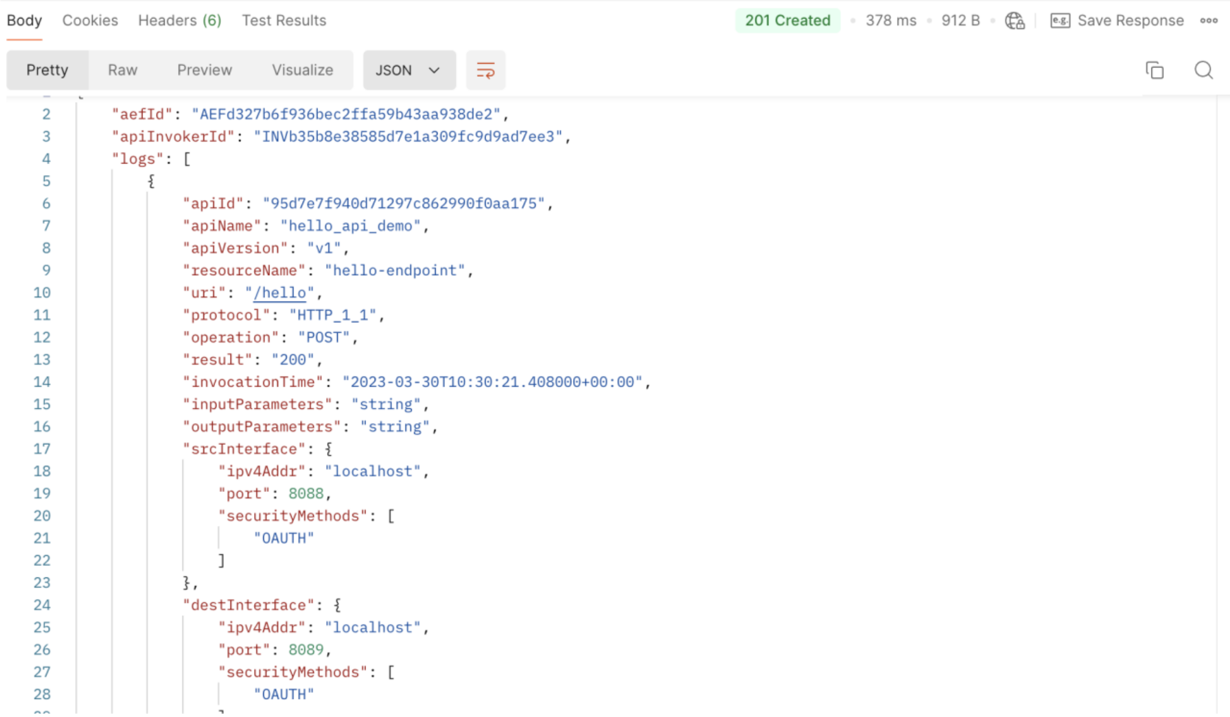
POST https:// {{CAPIF_HOSTNAME}} : {{CAPIF_PORT}} /api-invocation-logs/v1/ {{AEF_ID}} /logs

```
{
    "aefId": "{{AEF_ID}}",
    "apiInvokerId": "{{INVOKER_ID}}",
    "logs":[
        {
            "apiId": "{{API_ID}}",
            "apiName": "{{API_NAME}}",
            "apiVersion": "v1",
            "resourceName": "hello-endpoint",
            "uri": "/hello",
            "protocol": "HTTP_1_1",
            "operation": "POST",
            "result": "200",
            "invocationTime": "2023-03-30T10:30:21.408000+00:00",
            "invocationLatency": 0,
            "inputParameters": "string",
            "outputParameters": "string",
            "srcInterface": {
                "ipv4Addr": "localhost",
                "port": 8088,
                "securityMethods": [
                    "OAUTH"
                ]
            }
```

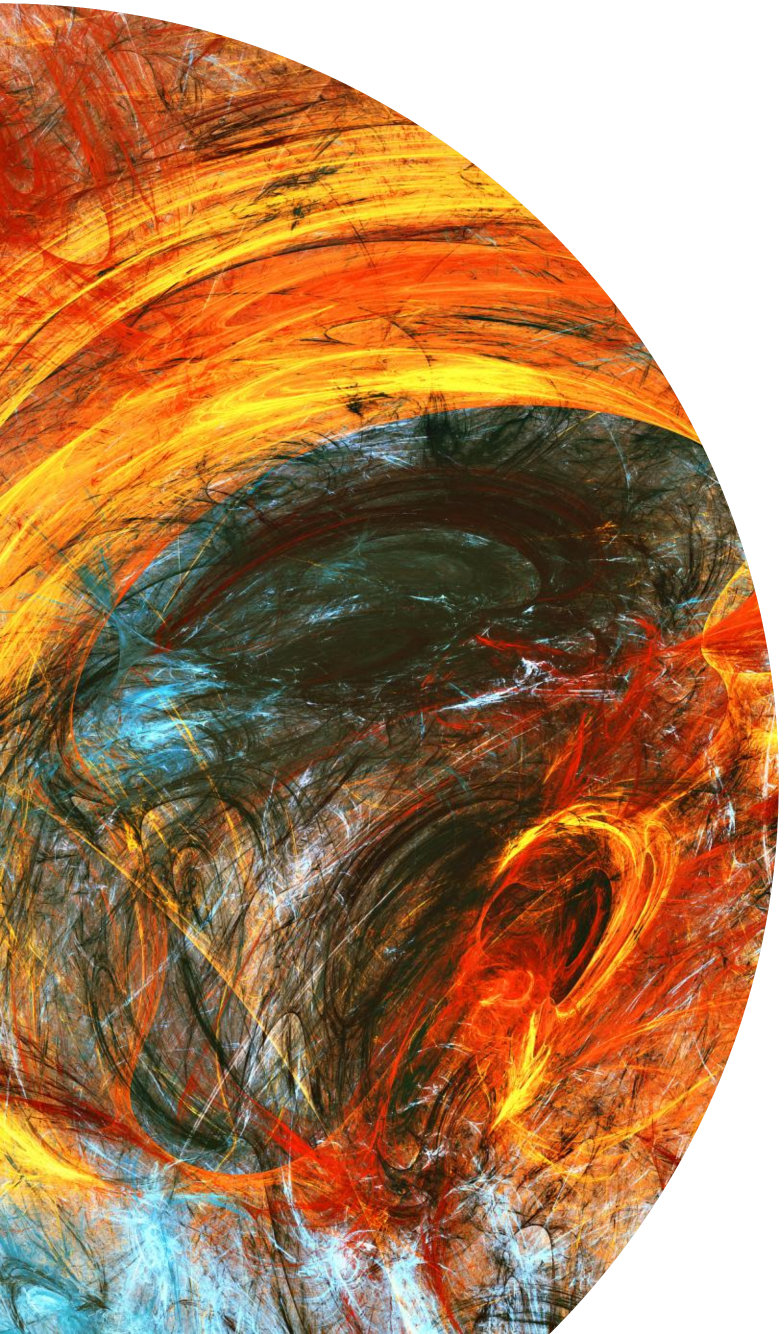


Create a Log by provider

Response received will be 201 Created with the Log saved in body:

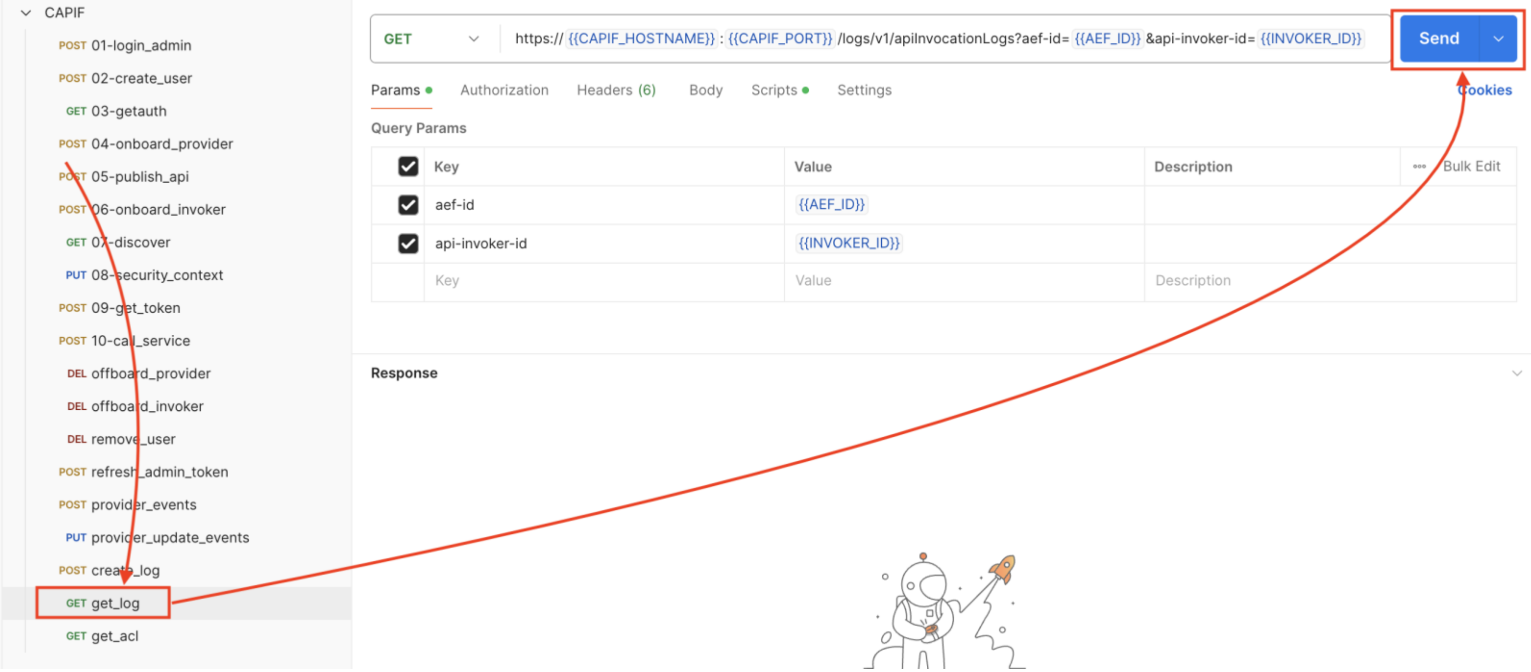


```
    "aefId": "AEFd327b6f936bec2ffa59b43aa938de2",
    "apiInvokerId": "INVb35b8e38585d7e1a309fc9d9ad7ee3",
    "logs": [
        {
            "apiId": "95d7e7f940d71297c862990f0aa175",
            "apiName": "hello_api_demo",
            "apiVersion": "v1",
            "resourceName": "hello-endpoint",
            "uri": "/hello",
            "protocol": "HTTP_1_1",
            "operation": "POST",
            "result": "200",
            "invocationTime": "2023-03-30T10:30:21.408000+00:00",
            "inputParameters": "string",
            "outputParameters": "string",
            "srcInterface": {
                "ipv4Addr": "localhost",
                "port": 8088,
                "securityMethods": [
                    "OAUTH"
                ]
            },
            "destInterface": {
                "ipv4Addr": "localhost",
                "port": 8089,
                "securityMethods": [
                    "OAUTH"
```



Get a Log by provider

Now we can select request get_log to use Auditing service:

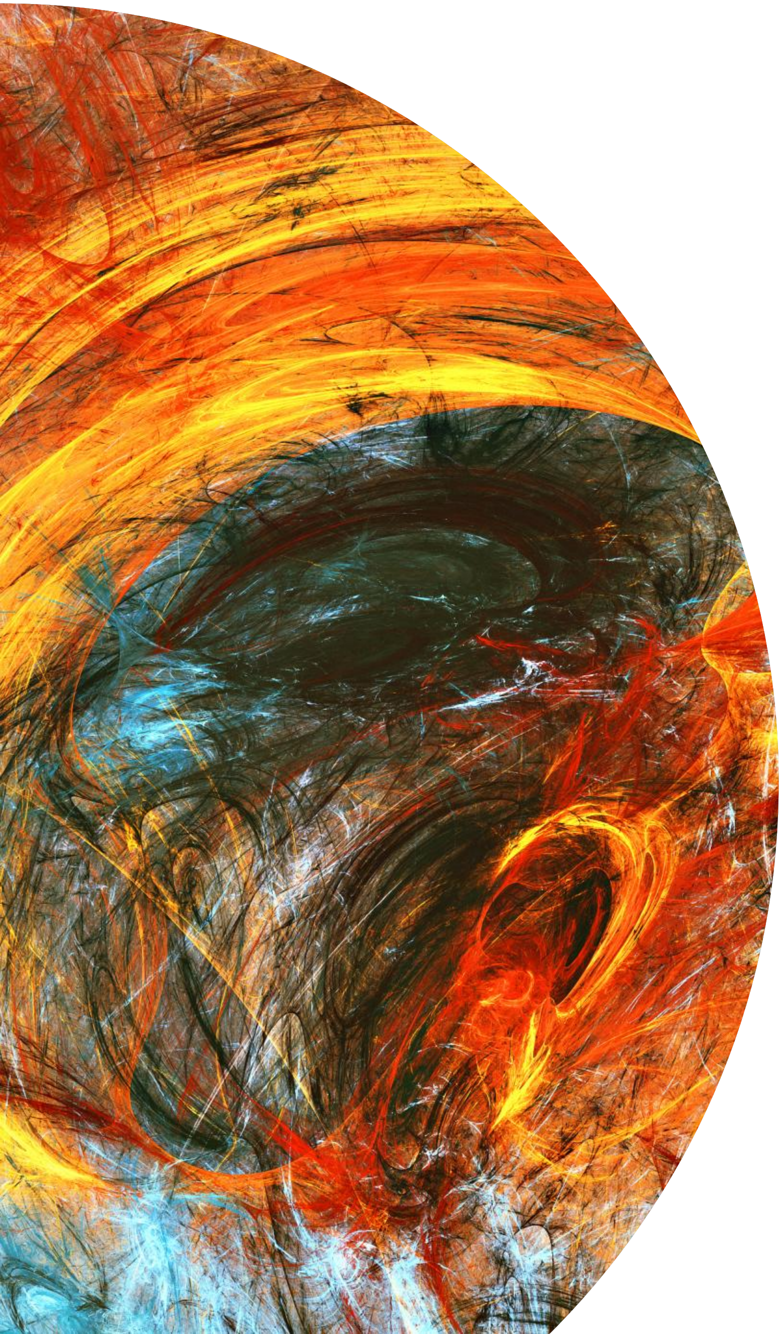


CAPIF
- POST 01-login_admin
- POST 02-create_user
- GET 03-getauth
- POST 04-onboard_provider
- POST 05-publish_api
- POST 06-onboard_invoker
- GET 07-discover
- PUT 08-security_context
- POST 09-get_token
- POST 10-call_service
- DEL offboard_provider
- DEL offboard_invoker
- DEL remove_user
- POST refresh_admin_token
- POST provider_events
- PUT provider_update_events
- POST create_log
- GET get_log
- GET get_acl

GET https:// {{CAPIF_HOSTNAME}} : {{CAPIF_PORT}} /logs/v1/apiInvocationLogs?aef-id= {{AEF_ID}} &api-invoker-id= {{INVOKER_ID}}

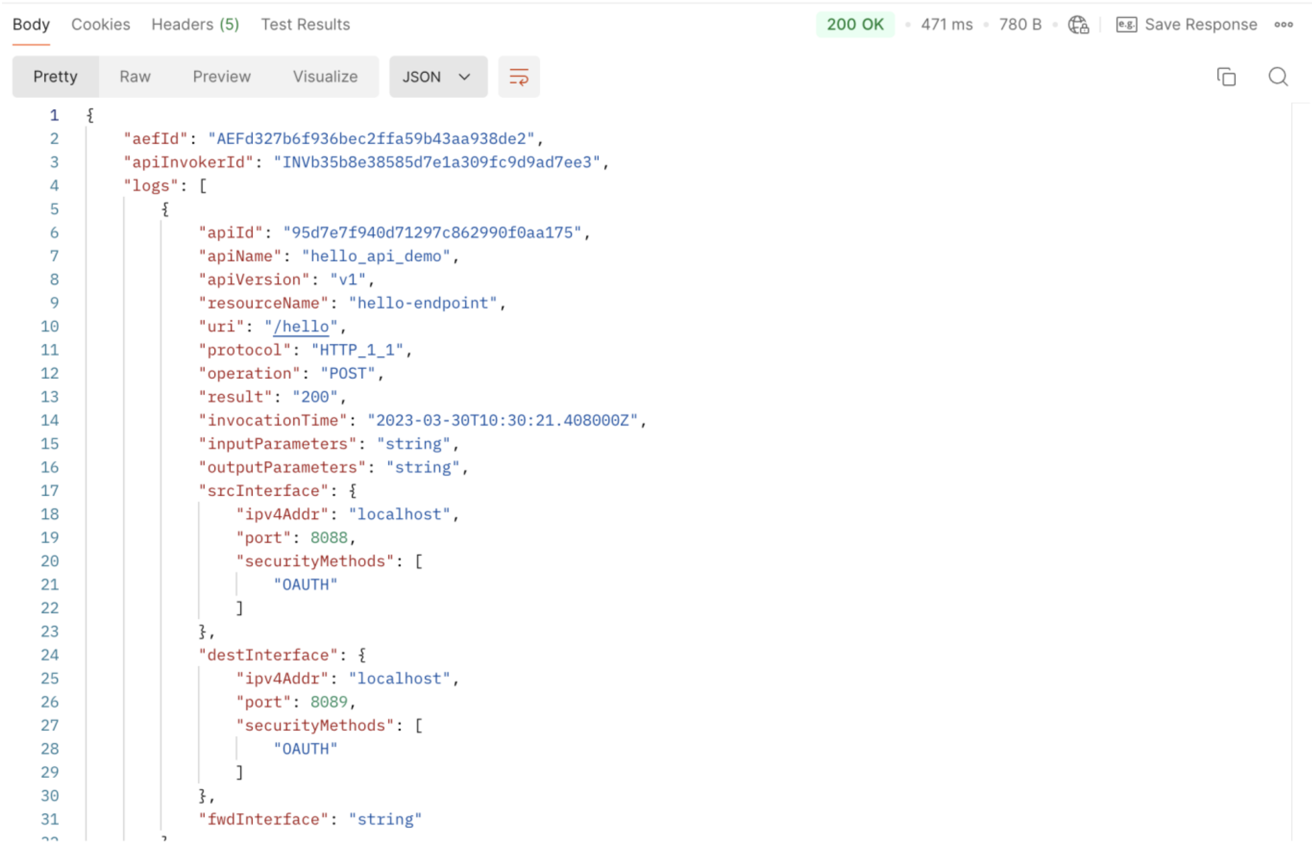
Query Params

	Key	Value	Description
☑	aef-id	{{AEF_ID}}	
☑	api-invoker-id	{{INVOKER_ID}}	
	Key	Value	Description

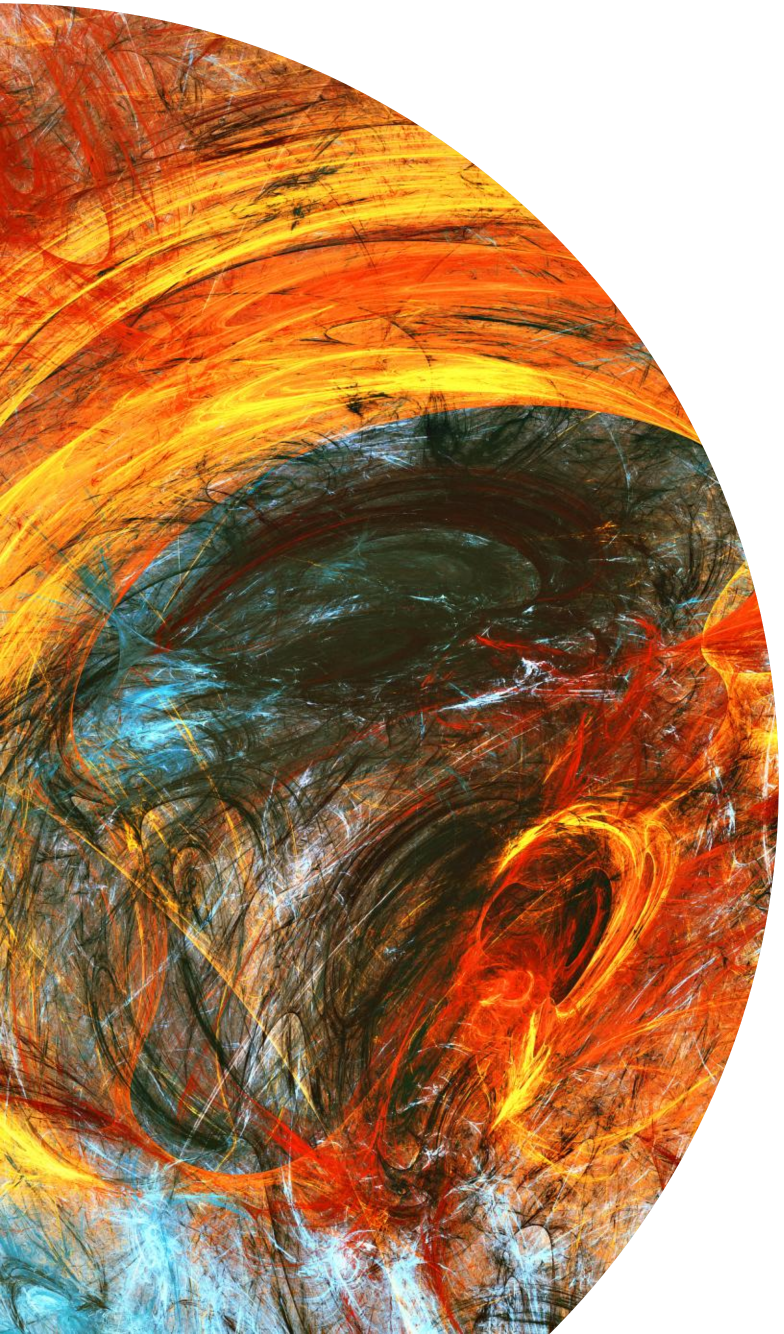


Get a Log by provider

Response received will be 200 OK with the Log that we saved previously in body:



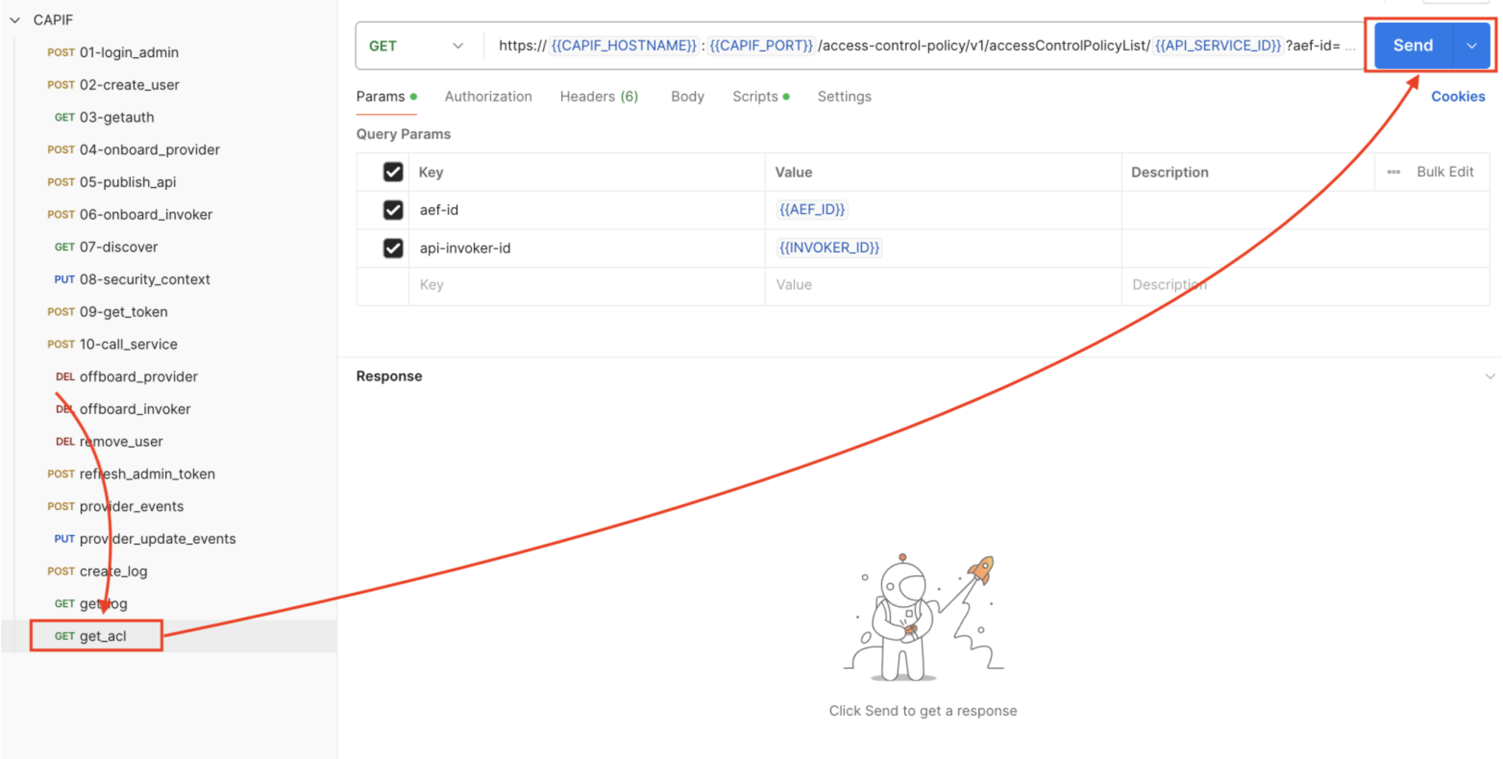
```json
{
    "aefId": "AEFd327b6f936bec2ffa59b43aa938de2",
    "apiInvokerId": "INVb35b8e38585d7e1a309fc9d9ad7ee3",
    "logs": [
        {
            "apiId": "95d7e7f940d71297c862990f0aa175",
            "apiName": "hello_api_demo",
            "apiVersion": "v1",
            "resourceName": "hello-endpoint",
            "uri": "/hello",
            "protocol": "HTTP_1_1",
            "operation": "POST",
            "result": "200",
            "invocationTime": "2023-03-30T10:30:21.408000Z",
            "inputParameters": "string",
            "outputParameters": "string",
            "srcInterface": {
                "ipv4Addr": "localhost",
                "port": 8088,
                "securityMethods": [
                    "OAUTH"
                ]
            },
            "destInterface": {
                "ipv4Addr": "localhost",
                "port": 8089,
                "securityMethods": [
                    "OAUTH"
                ]
            },
            "fwdInterface": "string"
```



Get an ACL by provider

OpenCAPIF

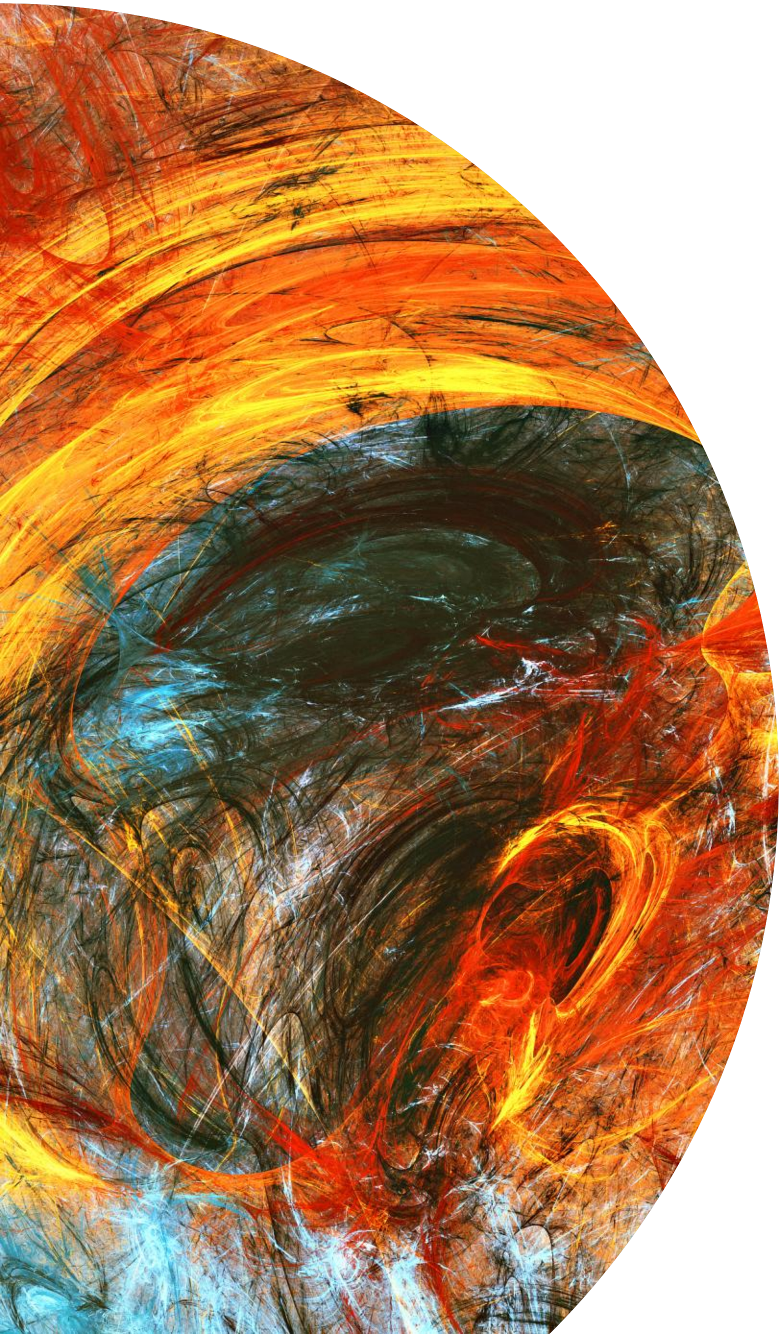
Now we can select request get_acl to recive the ACL of an Invoker:



GET https:// {{CAPIF_HOSTNAME}} : {{CAPIF_PORT}} /access-control-policy/v1/accessControlPolicyList/ {{API_SERVICE_ID}} ?aef-id= ...

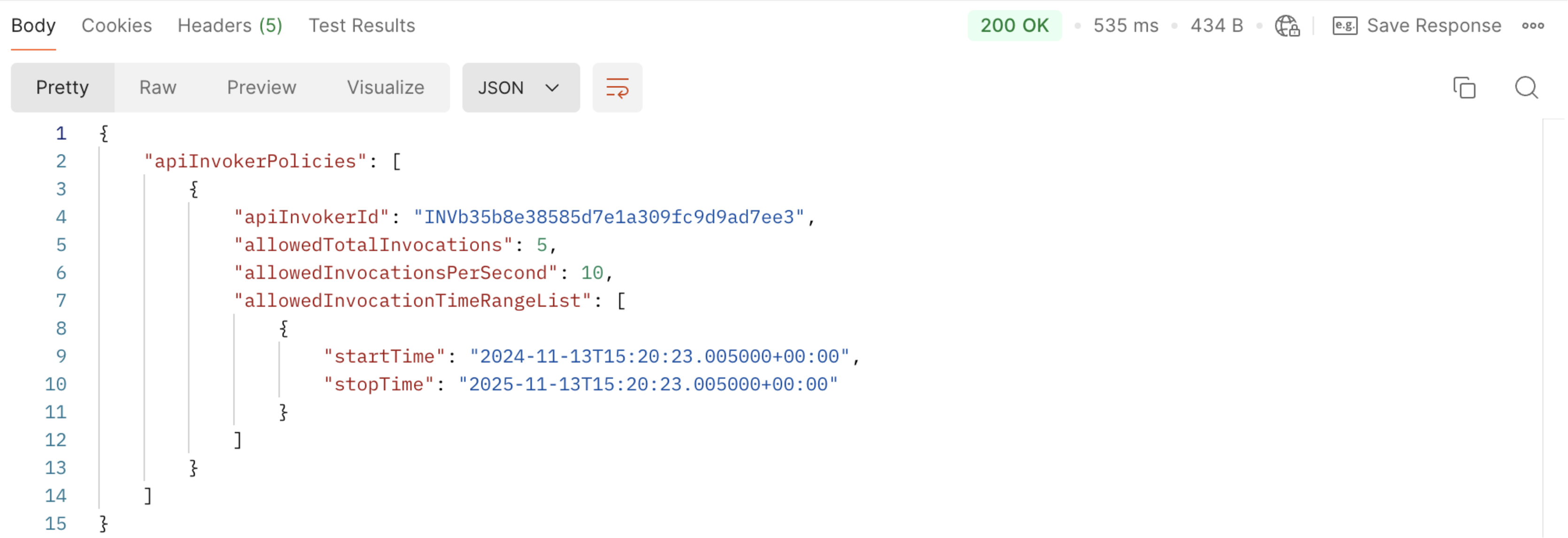
Query Params

☑	Key	Value	Description
☑	aef-id	{{AEF_ID}}	
☑	api-invoker-id	{{INVOKER_ID}}	
	Key	Value	Description



Get an ACL by provider

Response received will be 200 OK with th information of the Access Control Policy in body:



```json
{
    "apiInvokerPolicies": [
        {
            "apiInvokerId": "INVb35b8e38585d7e1a309fc9d9ad7ee3",
            "allowedTotalInvocations": 5,
            "allowedInvocationsPerSecond": 10,
            "allowedInvocationTimeRangeList": [
                {
                    "startTime": "2024-11-13T15:20:23.005000+00:00",
                    "stopTime": "2025-11-13T15:20:23.005000+00:00"
                }
            ]
        }
    ]
}
```