Agents and Functions Creation using APIs

Introduction

This guide shows how to createAgents andAgent Functions in Agentverse using APIs. With this guide, we set up a Python script that interacts with the Agentverse and help us creating Agents and Agent Functions. Before you begin, it is necessary that you satisfy the following requirements:

- Python version greater than 3.9 and less than 3.11.
- Therequests
- library is installed. You can install it using:pip install requests
- .
- Agentverse ↗(opens in a new tab)
- credentials.

How to get Agentverse API Tokens

- Go to Profile section inAgentverse ↗(opens in a new tab)
- .
- Click on button+ New API Key
- .
- Give name to your API key.
- Click onwrite
- forAccess to all resources in Agentverse
- and click onGenerate API Key

How to create Agents and respective Functions

1. Open terminal and create a directoryagents
2. using:mkdir agents
3. .
4. Create two Python files,agent.py
5. andagent_create.py
6. , in this directory and include the following sample scripts within them. You can do this using:touch agent.py
7. andtouch agent_create.py
8. .
9. Fill in the scripts with the code presented here below for each one of them:

Script 1:agent.py

import requests import json from ai_engine import UAgentResponse , UAgentResponseType

class

Coordinates (Model): location :

str

location_protocol

Protocol ("Location Coordinates")

async

def

location_coordinates (latitude ,

longitude): url =

"https://geocoding-by-api-ninjas.p.rapidapi.com/v1/reversegeocoding" querystring =

{ "lat" : latitude , "lon" : longitude }

headers

{ "X-RapidAPI-Key" :

"YOUR_API_KEY" , "X-RapidAPI-Host" :

"geocoding-by-api-ninjas.p.rapidapi.com" }

response

requests . get (url, headers = headers, params = querystring)

data

response . json () [0]['name']

return data

@location_protocol . on_message (model = Coordinates, replies = UAgentResponse) async

def

location_coordinates_calculator (ctx : Context ,

sender :

str ,

msg : Coordinates): ctx . logger . info (msg.location) latitude , longitude =

map (str .strip, msg.location. split (',')) city =

location_coordinates (latitude, longitude) ctx . logger . info (city) message = city await ctx . send (sender, UAgentResponse (message = message, type = UAgentResponseType.FINAL))

agent . include (location_protocol)

Script 2:agent_create.py

This script interacts with the Agentverse API to achieve the Agent and Function creation tasks.

Let's get started!

1. Import the required libraries and set up the authorization token:
2. # Importing Required libraries
3. import
4. time
5. import
6. requests
7. # Define access token
8. token
9. =
10. 'Bearer '
11. Take the agent name from user and store the agent's address:
12. # Take name of agent from user
13. name
14. =
15. input
16. (
17. 'Please give name of your agent? '

18.)

19. ## Create payload for agent creation request

20. agent_creation_data
21. =
22. {
23. "name"
24. :
25. name
26. }

27. ## Post request to create an agent and store address

28. response_agent
29. =
30. requests
31. .
32. post
33. (
34. "https://agentverse.ai/v1/hosting/agents"
35. , json
36. =
37. agent_creation_data, headers
38. =
39. {
40. "Authorization"
41. : token
42. }).
43. json
44. ()
45. address
46. =
47. response_agent
48. [
49. 'address'
50.]
51. print
52. (
53. f
54. 'Agent Address :
55. {
56. address
57. }
58. '
59.)
60. Take code fromagent.py
61. file. Then, store it in a dedicated script for the created agent:

62. ## Reading code to be placed in agent

63. with
64. open
65. (
66. 'agent.py'
67. ,
68. 'r'
69.)
70. as
71. file
72. :
73. code
74. =
75. file
76. .

77. read
78. ()
79. agent_code_data
80. =
81. {
82. "code"
83. :
84. code
85. }

86. Creating agent.py script for created agent

87. response_code_update
88. =
89. requests
90. .
91. put
92. (
93. f
94. "https://agentverse.ai/v1/hosting/agents/
95. {
96. address
97. }
98. /code"
99. , json
100. =
101. agent_code_data, headers
102. =
103. {
104. "Authorization"
105. : token
106. })

107. Starting the agent

108. requests
109. .
110. post
111. (
112. f
113. "https://agentverse.ai/v1/hosting/agents/
114. {
115. address
116. }
117. /start"
118. , headers
119. =
120. {
121. "Authorization"
122. : token
123. })
124. time
125. .
126. sleep
127. (
128. 10
129.)

130. waiting before getting agent's protocol

131. Requesting protocol digest for the created Agent:

132. Request to get agent protocol digest

133. response_protcol
134. =
135. requests
136. .
137. get
138. (
139. f
140. "https://agentverse.ai/v1/almanac/agents/
141. {
142. address
143. }
144. "
145. , headers
146. =
147. {
148. "Authorization"
149. : token
150. })
151. protocol_digest
152. =
153. response_protcol
154. .
155. json
156. ()
157. [
158. 'protocols'
159.][
160. 1
161.]
162. print
163. (
164. f
165. 'Protocol Digest :
166. {
167. protocol_digest
168. }
169. '
170.)
171. time
172. .
173. sleep
174. (
175. 10
176.)

177. Waiting before getting model_digest

178. RequestModel
179. digest and name using Almanac APIs:

180. Request to get agent's model details

181. response_model
182. =
183. requests
184. .
185. get
186. (
187. f
188. "https://agentverse.ai/v1/almanac/manifests/protocols/
189. {
190. protocol_digest
191. }
192. "
193. , headers
194. =

195. {
196. "Authorization"
197. : token
198. })
199. model
200. =
201. response_model
202. .
203. json
204. ()
205. [
206. 'models'
207.]
208. time
209. .
210. sleep
211. (
212. 10
213.)

214. Waiting before storing details to create services

215. Now, save all the required details to create the Agent Function. Then, create the Agent Function on the basis of the details received:

216. Taking inputs from user for details required to create a service

217. name_service
218. =
219. input
220. (
221. 'Please give service name'
222.)
223. description
224. =
225. input
226. (
227. 'Please enter service description'
228.)
229. field_name
230. =
231. input
232. (
233. 'Please enter field name'
234.)
235. field_description
236. =
237. input
238. (
239. 'Please enter field description'
240.)
241. tasktype
242. =
243. input
244. (
245. 'Please tell task or subtask'
246.)

247. Logging details provided by user

248. print
249. (
250. f

```
'Service name:
{
name_service
}
\n
Service Description:
{
description
}
\n
Field Name:
{
field_name
}
\n
Field Description:
{
field_description
}
\n
Task Type:
{
tasktype
}
'
)
```

Storing model digest and name to be used for service creation

```
model_digest
=
response_model
.
json
()
[
'interactions'
][
0
][
'request'
]
.
replace
(
'model:'
,
''
)
print
(
f
'Model Digest :
{
model_digest
}
'
)
model_name
=
model
[
0
]
[
'schema'
][
'title'
]
print
(
f
'Model Name :
{
model_name
}
'
)
```

Creating payload for service creation

```
data
=
{
"agent"
:
address
,
"name"
:
name_service
,
"description"
:
description
,
"protocolDigest"
:
protocol_digest
,
"modelDigest"
:
model_digest
,
"modelName"
:
model_name
,
"fields"
:
[
{
"name"
:
field_name
,
"required"
:
True
,
"field_type"
:
"string"
,
"description"
:
field_description
}
]
,
"taskType"
:
```

379. tasktype
380. }
381. **Requesting AI Engine services API to create a service with created payload and storing the response.**
382. response_service
383. =
384. requests
385. .
386. post
387. (
388. "https://agentverse.ai/v1beta1/services"
389. , json
390. =
391. data, headers
392. =
393. {
394. "Authorization"
395. : token
396. })
397. **Storing name of function and printing it to check if function was created successfully**
398. name
399. =
400. response_service
401. .
402. json
403. ()
404. [
405. 'name'
406.]
407. print
408. (
409. f
410. 'Service Created with name:
411. {
412. name
413. }
414. '
415.)

Complete Script

Importing Required libraries

import time import requests

Decode the refresh token

token

f 'Bearer '

Take name of agent from user

name

input ('Please give name of your agent? ')

Create payload for agent creation request

agent_creation_data

{ "name" : name }

Post request to create an agent and store address

response_agent

requests . post ("https://agentverse.ai/v1/hosting/agents" , json = agent_creation_data, headers = { "Authorization" : token}). json ()

address

response_agent ['address'] print (f 'Agent Address : { address } ')

Reading code to be placed in agent

with

open ('agent.py' , 'r')

as file : code = file . read () agent_code_data =

{ "code" : code }

Creating agent.py script for created agent

response_code_update

requests . put (f "https://agentverse.ai/v1/hosting/agents/ { address } /code" , json = agent_code_data, headers = { "Authorization" : token})

Starting the agent

requests . post (f "https://agentverse.ai/v1/hosting/agents/ { address } /start" , headers = { "Authorization" : token}) time . sleep (10)

waiting before getting agent's protocol

Request to get agent protocol digest

response_protcol

requests . get (f "https://agentverse.ai/v1/almanac/agents/ { address } " , headers = { "Authorization" : token}) protocol_digest = response_protcol . json () ['protocols'][1] print (f 'Protocol Digest : { protocol_digest } ') time . sleep (10)

Waiting before getting model_digest

Request to get agent's model details

response_model

requests . get (f "https://agentverse.ai/v1/almanac/manifests/protocols/ { protocol_digest } " , headers = { "Authorization" : token}) model = response_model . json () ['models'] time . sleep (10)

Waiting before storing details to create services

Taking inputs from user for details required to create a function

name_service

input ('Please give service name') description =

input ('Please enter service description') field_name =

input ('Please enter field name') field_description =

input ('Please enter field description') tasktype =

input ('Please tell task or subtask')

Logging details provided by user

print (f 'Service name: { name_service }

\n Service Description: { description }

\n Field Name: { field_name } \n Field Description: { field_description } \n Task Type: { tasktype } ')

Storing model digest and name to be used for fucntion creation

model_digest

response_model . json () ['interactions'][0]['request'] . replace ('model:' , '') print (f 'Model Digest : { model_digest } ') model_name = model [0] ['schema']['title'] print (f 'Model Name : { model_name } ')

Creating payload for function creation

data

{ "agent" : address , "name" : name_service , "description" : description , "protocolDigest" : protocol_digest , "modelDigest" : model_digest , "modelName" : model_name , "fields" : [{ "name" : field_name , "required" :

True , "field_type" :

"string" , "description" : field_description }] , "taskType" : tasktype }

Requesting AI Engine functions API to create a function

with created payload and storing the response.

response_service

requests . post ("https://agentverse.ai/v1beta1/services" , json = data, headers = { "Authorization" : token })

Storing name of function and printing it to check if function was created successfully

name

response_service . json () ['name'] print (f 'Service Created with name: { name } ')

Steps to run the script

1. Open terminal and go to directoryagents
2. created above.
3. Make sureagent.py
4. andagent_create.py
5. are in this directory.
6. Head over to theAgentverse ↗(opens in a new tab)
7. andgenerate API keys ↗
8. .
9. Open script in editor and replacetoken
10. field.
11. Run commandpython agent_create.py
12. and enter the required details.
13. Provide Agent and Function Details as asked, then, check agent and function on Agentverse.

Expected Output

- Provide all details asked in the script:

- Agent created on Agentverse:

- Function created on Agentverse:

Was this page helpful?

AVCTL Hosting commands Secret Management APIs