Hosted, Local, and Mailbox Agents

Agents can operate in multiple environments based on how they are created and deployed within the network. Understanding the difference betweenHosted ,Local , andMailbox Agents is helpful for developers to correctly choose the right setup for their use cases and applications.

Hosted Agents

Hosted Agents are cloud-based agents managed within theAgentverse(opens in a new tab) . This allows developers to build, deploy, and run Agents without worrying about infrastructure management. The Agentverse ensures that your agents run as functions, executing routines and handling messages straightforwardly in response to specific triggers. This design keeps the operational costs of hosted agents low, which is passed on to the users.

Agentverse provides a variety of tools to make agent development easy. First of all, the Agent Editor which allows you to write, edit, and manage your agents directly in the Agentverse environment. It includes essential features such as the Agent Logs, where you can view real-time outputs of your agent's execution, helping you monitor and debug your agents on the fly. The platform also offers storage and secrets management, enabling you to securely manage data and information. One important feature of hosted agents ishow they manage their state . Since hosted agents do not run continuously, any global variable you define will always revert to its initialized value after each function call. For example, if you try to increment a global counter, its value will reset to the starting value every time the function is invoked. To persist data between calls, you must use the Agent Storage; this ensures key values are maintained across function executions. This is crucial for developing agents that require stateful behavior over time. .

When creating hosted agents, you can either start from a blank script or use one of the many predefined templates available. These templates cover a range of use cases, such as smart services or AI/ML integrations, allowing developers to quickly get started with agents designed to toss a coin, retrieve stock prices, or find nearby restaurants. The template system simplifies agent creation by providing a foundation, while also allowing for customization to meet specific requirements following the idea users have in mind.

A key advantage of hosted agents is that they do not require local dependencies, making development more accessible. The system provides an integratedagent.py file, and although hosted agents can be created based on a restricted set of imports, they still offer significant flexibility for developers to create complex solutions.

For a better reference to these topics, check out the following resources:

- Creating Hosted Agents
- .
- Agentverse: Allowed Imports
- .
- How Agents handle their state
- .

Local Agents

Local agents instead run directly on your machine or infrastructure; this gives you full control over environment, configuration, and execution. Unlike Agentverse hosted agents, local agents can continuously run processes that handle events, messages, and tasks in real-time without relying on an orchestrator. This setup allows for enhanced flexibility and dynamic behavior, as agents can manage their state across function calls without needing to store data externally. This feature makes them ideal for use cases where real-time processing and direct access to local resources are critical.

When developing local agents, you can leverage the entire uAgents Framework, which offers a wide range of tools to support agent-based development. Also, since local agents do not face the same constraints as hosted agents, you are free to use any Python package or custom module that fits your project; this increases flexibility in design, and enables the use of powerful libraries for machine learning, data processing, and external integrations that may not be supported on Agentverse.

A key benefit of local agents is their ability to maintain persistent state without external storage. Since they run continuously, global variables and internal states are maintained across multiple function executions. For instance, if you have a counter that increments every time a message is received, it will correctly retain its value throughout the agent's lifecycle. This contrasts with hosted agents, where global variables are re-initialized with each call and require storage to persist data.

Running a local agent involves setting up a Python environment on your machine, installing the necessary dependencies, and executing the agent script. Agents can run indefinitely, waiting for incoming messages or scheduled tasks, which makes them well-suited for applications that require constant monitoring, immediate response handling, or interactions with other agents in real-time.

A local agent can also be run within a Docker container, making it easier to deploy agents in a production environment or across distributed systems. This is particularly useful when building functions that need to scale across multiple systems or require isolation from other processes.

Local agents allow for deeper customization and optimization features; you can manage resource allocation, tune performance settings, and integrate directly with other services running on the same machine. This makes local agents the preferred choice for high-performance applications or scenarios where you need to closely manage system resources and agent behavior.

Head over to the following resource for a better understanding of Agents and their applications:

- Agents - uAgents Framework
- .
- Creating your first agent
- .
- Agent Functions
- .
- Options for running your Agents
- .

Mailbox Agents

TheAgentverse Mailbox feature makes agents a hybrid between Hosted and Local Agents. This because Local Agents may not always be online all the time due to network outages, being behind a firewall, or intentionally going offline for resource management. It may also be that you need to create an Agent application using imports and tools that are currently not available in the Agentverse, and thus you need to design agents locally to guarantee fully operative and runnable application. To manage such scenarios, theMailbox feature in Agentverse allows agents to receive messages while they are offline with ease. Once the agent comes back online, it can retrieve these messages from its mailbox.

Local agents can use a Mailbox to ensure that no messages are lost when they are temporarily disconnected from the network; the Mailbox acts as a message buffer, storing communications until the agent comes back online and ready to process them. Indeed, this feature enables interaction with other agents or functions without the agent being online continuously.

In order to set up a mailbox for a local agent, you first need to create and configure a local agent. The agent's address is used to register it on the Agentverse, and then to generate aMailbox API key for your agent. For instance, consider the following basic agent:

Self hosted mailbox_agent.py from uagents import Agent , Context , Model

class

Message (Model): message :

str

AGENT_MAILBOX_KEY

"put_your_AGENT_MAILBOX_KEY_here" SEED_PHRASE =

"put_your_seed_phrase_here"

Now your agent is ready to join the agentverse!

agent

Agent (name = "alice" , seed = SEED_PHRASE, mailbox = f " { AGENT_MAILBOX_KEY } @https://agentverse.ai" ,)

Copy the address shown below

print (f "Your agent's address is: { agent.address } ")

if

name

==

"**main**" : agent . run () Then, head over to theAgentverse(opens in a new tab) and navigate to theMy Agents tab. Here, selectLocal Agents and click onConnect Local Agent . Paste your agent's address retrieved from running your local agent and follow the steps until you get aMailbox API Key , which you will use to update your agent's configuration above. To test your Mailbox setup, you can create another agent (on Agentverse for instance) that sends messages to the Mailbox while the first agent is offline. When the first agent comes back online, it will retrieve and process the stored messages. For a complete example, check out thisguide .

For a more complex example of an agent using a Mailbox, check out the followinguide .

Last updated on October 17, 2024

Was this page helpful?

You can also leave detailed feedbackon Github

Send tokens with Agents Agents Mailboxes