[crewAI] Agents

Core Concept	
1. Agents	An agent is an **autonomous unit** programmed to: • Perform tasks • Make decisions • Communicate with other agents Think of an agent as a member of a team, with specific skills and a particular job to do. Agents can have different roles like 'Researcher', 'Writer', or 'Customer Support', each contributing to the overall goal of the crew.

Agent Attributes

Attribute	Parameter	Description
Role	`role`	Defines the agent's function within the crew. It determines the kind of tasks the agent is best suited for.
Goal	`goal`	The individual objective that the agent aims to achieve. It guides the agent's decision-making process.
Backstory	`backstory`	Provides context to the agent's role and goal, enriching the interaction and collaboration dynamics.
LLM *(optional)*	`llm`	Represents the language model that will run the agent. It dynamically fetches the model name from the `OPENAI_MODEL_NAME` environment variable, defaulting to "gpt-4" if not specified.
Tools *(optional)*	`tools`	Set of capabilities or functions that the agent can use to perform tasks. Expected to be instances of custom classes compatible with the agent's execution environment. Tools are initialized with a default value of an empty list.
Function Calling LLM *(optional)*	`function_calling_llm`	Specifies the language model that will handle the tool calling for this agent, overriding the crew function calling LLM if passed. Default is `None`.
Max Iter *(optional)*	`max_iter`	Max Iter is the maximum number of iterations the agent can perform before

		being forced to give its best answer. Default is `25`.
Max RPM *(optional)*	`max_rpm`	Max RPM is the maximum number of requests per minute the agent can perform to avoid rate limits. It's optional and can be left unspecified, with a default value of `None`.
Max Execution Time *(optional)*	`max_execution_time`	Max Execution Time is the maximum execution time for an agent to execute a task. It's optional and can be left unspecified, with a default value of `None`, meaning no max execution time.
Verbose *(optional)*	`verbose`	Setting this to `True` configures the internal logger to provide detailed execution logs, aiding in debugging and monitoring. Default is `False`.
Allow Delegation *(optional)*	`allow_delegation`	Agents can delegate tasks or questions to one another, ensuring that each task is handled by the most suitable agent. Default is `True`.
Step Callback *(optional)*	`step_callback`	A function that is called after each step of the agent. This can be used to log the agent's actions or to perform other operations. It will overwrite the crew `step_callback`.
Cache *(optional)*	`cache`	Indicates if the agent should use a cache for tool usage. Default is `True`.
System Template *(optional)*	`system_template`	Specifies the system format for the agent. Default is `None`.
Prompt Template *(optional)*	`prompt_template`	Specifies the prompt format for the agent. Default is `None`.
Response Template *(optional)*	`response_template`	Specifies the response format for the agent. Default is `None`.
Allow Code Execution *(optional)*	`allow_code_execution`	Enable code execution for the agent. Default is `False`.
Max Retry Limit *(optional)*	`max_retry_limit`	Maximum number of retries for an agent to execute a task when an error occurs. Default is `2`.

Creating an Agent	
Agent Interaction	Agents can interact with each other using crewAI's built-in delegation and communication mechanisms. This allows for dynamic task management and problem-solving within the crew.

To create an agent, you would typically initialize an instance of the `Agent` class with the desired properties. Here's a conceptual example including all attributes:

```python
# Example: Creating an agent with all attributes
from crewai import Agent

agent = Agent(
  role='Data Analyst',
  goal='Extract actionable insights',
  backstory="""You're a data analyst at a large company.
  You're responsible for analyzing data and providing insights
  to the business.
  You're currently working on a project to analyze the
  performance of our marketing campaigns.""",
  tools=[my_tool1, my_tool2],  # Optional, defaults to an empty list
  llm=my_llm,  # Optional
  function_calling_llm=my_llm,  # Optional
  max_iter=15,  # Optional
  max_rpm=None, # Optional
  max_execution_time=None, # Optional
  verbose=True,  # Optional
  allow_delegation=True,  # Optional
  step_callback=my_intermediate_step_callback,  # Optional
  cache=True,  # Optional
  system_template=my_system_template,  # Optional
  prompt_template=my_prompt_template,  # Optional
  response_template=my_response_template,  # Optional
  config=my_config,  # Optional
  crew=my_crew,  # Optional
  tools_handler=my_tools_handler,  # Optional
  cache_handler=my_cache_handler,  # Optional
  callbacks=[callback1, callback2],  # Optional
  allow_code_execution=True,  # Optiona
  max_retry_limit=2,  # Optional
)
```

Setting prompt templates

Prompt templates	Prompt templates are used to format the prompt for the agent. You can use to update the system, regular and response templates for the agent. Here's an example of how to set prompt templates:

```python
# Example: Creating an agent with all attributes
from crewai import Agent

agent = Agent(
  role='Data Analyst',
  goal='Extract actionable insights',
  backstory="""You're a data analyst at a large company.
  You're responsible for analyzing data and providing insights
  to the business.
  You're currently working on a project to analyze the
  performance of our marketing campaigns.""",
  tools=[my_tool1, my_tool2],  # Optional, defaults to an empty list
  llm=my_llm,  # Optional
  function_calling_llm=my_llm,  # Optional
  max_iter=15,  # Optional
  max_rpm=None, # Optional
  max_execution_time=None, # Optional
  verbose=True,  # Optional
  allow_delegation=True,  # Optional
  step_callback=my_intermediate_step_callback,  # Optional
  cache=True,  # Optional
  system_template=my_system_template,  # Optional
  prompt_template=my_prompt_template,  # Optional
  response_template=my_response_template,  # Optional
  config=my_config,  # Optional
  crew=my_crew,  # Optional
  tools_handler=my_tools_handler,  # Optional
  cache_handler=my_cache_handler,  # Optional
  callbacks=[callback1, callback2],  # Optional
  allow_code_execution=True,  # Optiona
  max_retry_limit=2,  # Optional
)
```

Third Party Agents

Extend your Third Party Agents like LlamaIndex, Langchain, Autogen or fully custom agents using the the crewai's BaseAgent class.

BaseAgent includes attributes and methods required to integrate with your crews to run and delegate tasks to other agents within your own crew.

CrewAI is a universal multi agent framework that allows for all agents to work together to automate tasks and solve problems.

```python
from crewai import Agent, Task, Crew
from custom_agent import CustomAgent # You need to build and extend your own agent logic with the CrewAI BaseAgent class then import it here.

from langchain.agents import load_tools

langchain_tools = load_tools(["google-serper"], llm=llm)

agent1 = CustomAgent(
    role="agent role",
    goal="who is {input}?",
    backstory="agent backstory",
    verbose=True,
)

task1 = Task(
    expected_output="a short biography of {input}",
    description="a short biography of {input}",
    agent=agent1,
)

agent2 = Agent(
    role="agent role",
    goal="summarize the short bio for {input} and if needed do more research",
    backstory="agent backstory",
    verbose=True,
)

task2 = Task(
    description="a tldr summary of the short biography",
    expected_output="5 bullet point summary of the biography",
    agent=agent2,
    context=[task1],
)

my_crew = Crew(agents=[agent1, agent2], tasks=[task1, task2])
crew = my_crew.kickoff(inputs={"input": "Mark Twain"})
```