Getting Started with CrewAI: A Developer's Guide

This tutorial provides a clear, detailed, and graphical guide for developers setting up CrewAI, covering installation, project creation, and execution.

System Requirements & Dependency Management

CrewAI requires Python versions between 3.10 and 3.14. Verify your current version with

```
python3 --version
```

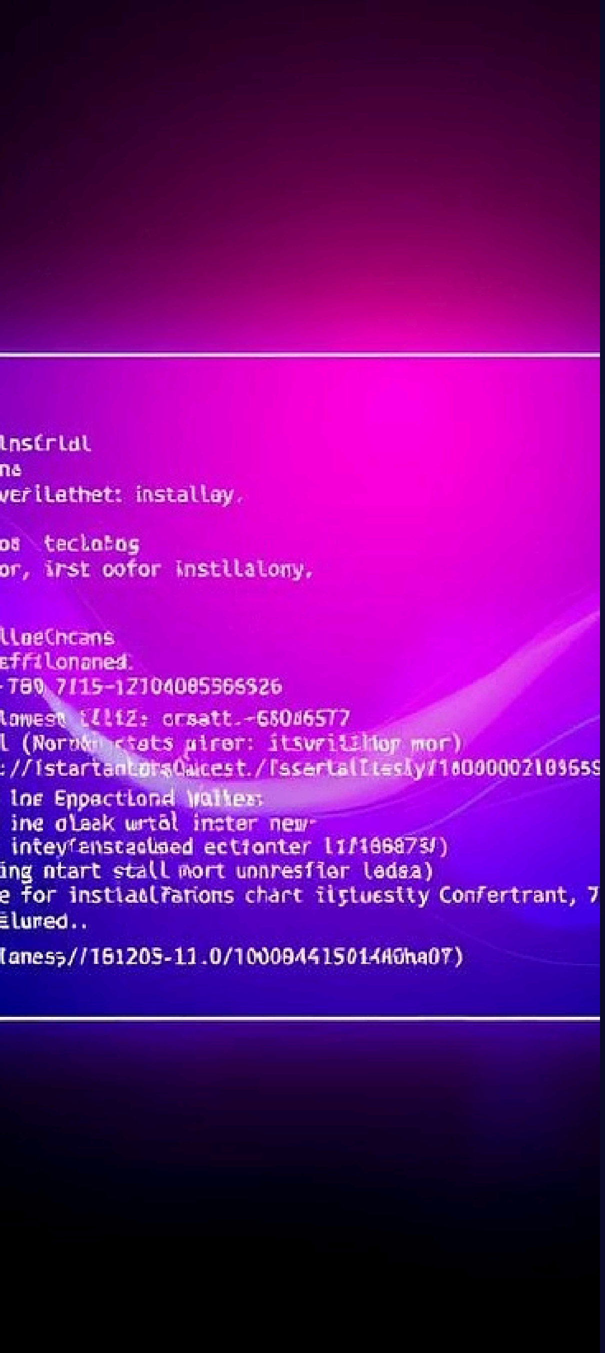
. If an update is needed, visit **python.org/downloads**

Python Version

Ensure you have Python >=3.10 and <3.14 installed. This range ensures compatibility with CrewAI's features and dependencies.

Dependency Manager

CrewAI utilizes uv for streamlined dependency management. It's a robust tool that handles package installation and project environments efficiently.



Step 1: Install uv

Installing uv is a straightforward process, depending on your operating system. Choose the command that suits your environment.

macOS/Linux Installation

For macOS and Linux users, uv can be installed via **curl** or **wget**. Open your terminal and run one of the following commands:

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

Alternatively, if curl is not available:

```
wget -qO- https://astral.sh/uv/install.sh | sh
```

Windows Installation

Windows users can install uv using **PowerShell**. Launch PowerShell as an administrator and execute this command:

```
powershell -ExecutionPolicy ByPass -c "irm
https://astral.sh/uv/install.ps1 | iex"
```

For any installation issues, consult UV's official guide.

Step 2: Install CrewAI

With uv installed, you can now proceed to install the CrewAI CLI. This command will set up the necessary tools to begin developing your AI crews.

Installation Command

Run the following command in your terminal to install the CrewAI CLI:

```
uv tool install crewai
```

If you encounter a PATH warning, update your shell by running:

```
uv tool update-shell
```

Troubleshooting & Verification

Windows users might face a build error (`chroma-hnswlib==0.7.6`). This can be resolved by installing **Visual Studio Build Tools** with *Desktop development with C++*.

To verify the installation, execute:

```
uv tool list
```

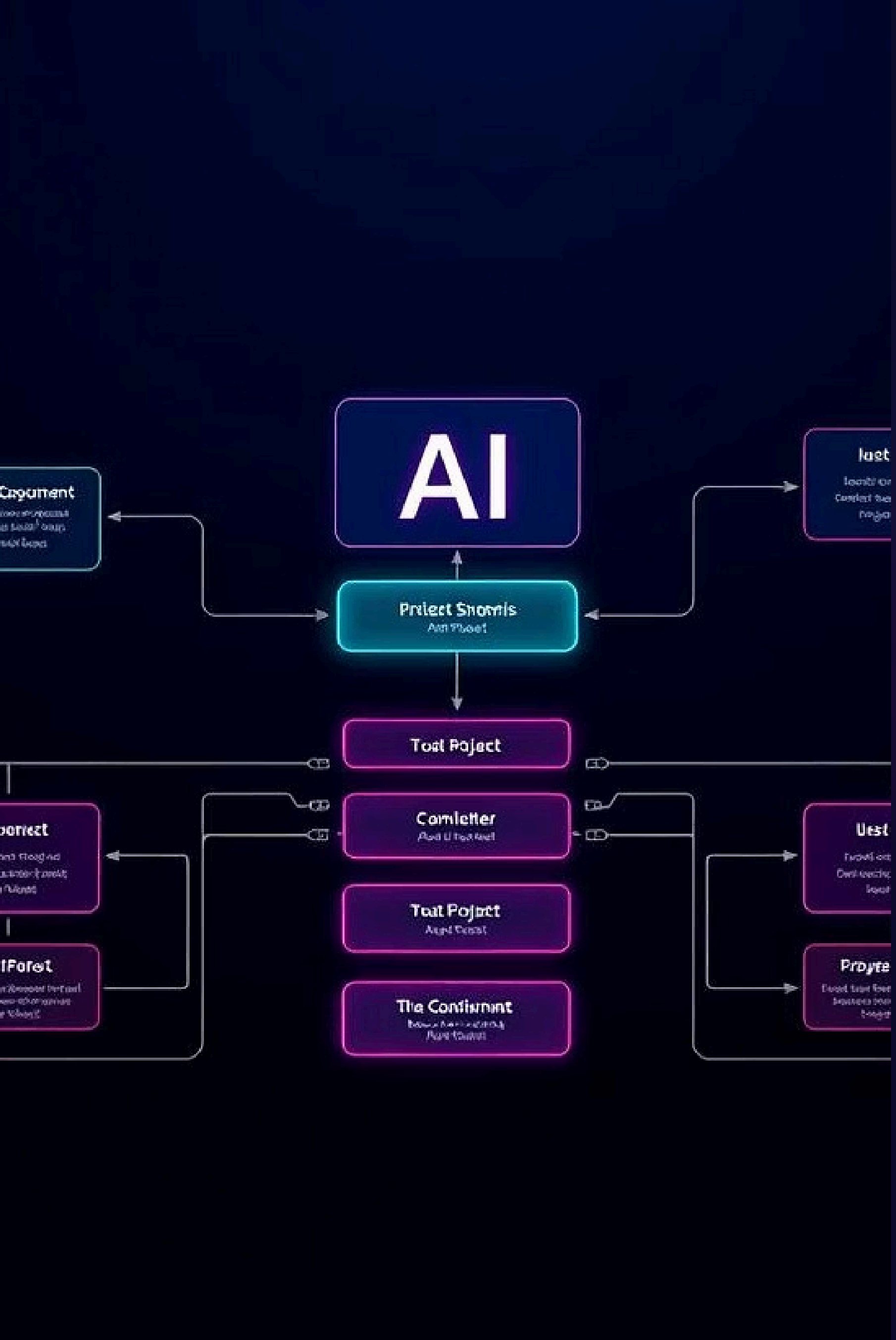
You should see:

```
crewai v0.102.0 - crewai
```

To update CrewAI, use:

```
uv tool install crewai --upgrade
```

Congratulations! CrewAI is now installed, and you're ready to create your first intelligent crew. 🎉



Creating a CrewAI Project

CrewAI simplifies project creation through YAML template scaffolding, providing a structured approach to define your agents and tasks. This method ensures a clear organization for your AI project.

Made with GAMMA

Step 1: Generate Project Scaffolding

To kickstart your CrewAI project, use the CLI command to generate a new project scaffolding. This command creates a pre-defined directory structure, including all essential files for your crew.

Run the following command, replacing `<your_project_name>` with your desired project name:

```
crewai create crew <your_project_name>
```

This command will generate a comprehensive project structure:

```
my_project/
 ├─── .gitignore
 ├─── knowledge/
 ├─── pyproject.toml
 ├─── README.md
 ├─── .env
 └─── src/
  └─── my_project/
  ├─── __init__.py
  ├─── main.py
  ├─── crew.py
  ├─── tools/
  │   ├─── custom_tool.py
  │   └─── __init__.py
  └─── config/
  ├─── agents.yaml
  └─── tasks.yaml
```

Step 2: Customize Your Project

Once your project scaffolding is generated, you can customize the key files to define your crew's behavior, tasks, and environment settings.

agents.yaml	Define your AI agents and their roles, capabilities, and backstories.
tasks.yaml	Set up agent tasks and workflows, outlining their objectives and dependencies.
.env	Securely store API keys and other environment variables required for your project.
main.py	The primary entry point for your project and its execution flow.
crew.py	Handles the orchestration and coordination of your entire CrewAI crew.
tools/	A directory for custom agent tools, allowing you to extend functionality.
knowledge/	A dedicated directory for your crew's knowledge base and reference materials.

Begin by modifying `agents.yaml` and `tasks.yaml` to tailor your crew’s functionality. Remember to keep sensitive data like API keys in the `.env` file.

Step 3: Run Your Crew

With your project configured, you're ready to bring your AI crew to life. Follow these final steps to install dependencies and execute your CrewAI project.

Install Dependencies

Navigate to the root of your project directory and run:

```
crewai install
```

This command ensures all necessary packages are in place.

Add New Packages

If your project requires additional Python packages, use uv to install them:

```
uv add <package-name>
```

This keeps your environment up-to-date.

Execute Your Crew

Finally, to run your CrewAI project, execute the following command from your project's root directory:

```
crewai run
```

Your intelligent crew will now begin its operations!