(new-model-registration)=

Registering a Model to vLLM

vLLM relies on a model registry to determine how to run each model.
A list of pre-registered architectures can be found [here](#supported-models).

If your model is not on this list, you must register it to vLLM.
This page provides detailed instructions on how to do so.

Built-in models

To add a model directly to the vLLM library, start by forking our [GitHub repository](https://github.com/vllm-project/vllm) and then [build it from source](#build-from-source).
This gives you the ability to modify the codebase and test your model.

After you have implemented your model (see [tutorial](#new-model-basic)), put it into the <gh-dir:vllm/model_executor/models> directory.
Then, add your model class to `_VLLM_MODELS` in <gh-file:vllm/model_executor/models/registry.py> so that it is automatically registered upon importing vLLM.
Finally, update our [list of supported models](#supported-models) to promote your model!

:::{important}
The list of models in each section should be maintained in alphabetical order.
:::

Out-of-tree models

You can load an external model using a plugin without modifying the vLLM codebase.

:::{seealso}
[vLLM's Plugin System](#plugin-system)
:::

To register the model, use the following code:

```python
from vllm import ModelRegistry
from your_code import YourModelForCausalLM
ModelRegistry.register_model("YourModelForCausalLM", YourModelForCausalLM)
```

If your model imports modules that initialize CUDA, consider lazy-importing it to avoid errors like `RuntimeError: Cannot re-initialize CUDA in forked subprocess`:

```python
from vllm import ModelRegistry

ModelRegistry.register_model("YourModelForCausalLM", "your_code:YourModelForCausalLM")
```

:::{important}
If your model is a multimodal model, ensure the model class implements the

{class}`~vllm.model_executor.models.interfaces.SupportsMultiModal` interface.

Read more about that [here](#supports-multimodal).

:::

:::{note}

Although you can directly put these code snippets in your script using `vllm.LLM`, the recommended way is to place these snippets in a vLLM plugin. This ensures compatibility with various vLLM features like distributed inference and the API server.

:::