The `grid` geometry manager in Tkinter organizes widgets in a table-like structure. It divides the window or a Frame into rows and columns, allowing you to place widgets at specific positions within this grid. This method provides more control over the placement of widgets compared to the `pack` geometry manager, which organizes widgets in blocks before placing them in the parent widget.

Widgets can span multiple rows or columns, and the size of rows and columns can be set to change dynamically based on the content. The `grid` method is particularly useful for creating forms or any layout that requires precise positioning of widgets.

Key Options of `grid` Method:

- `column`: The column to put widget in; default is 0 (first column).
- `row`: The row to put widget in; default is 0 (first row).
- `columnspan`: How many columns widget occupies; default is 1.
- `rowspan`: How many rows widget occupies; default is 1.
- `sticky`: What to align the widget against within the grid cell. It uses compass directions as values (N, E, S, W, NE, NW, SE, SW). For example, `N` aligns the widget to the top center of the cell.

Example Code:

```
import tkinter as tk

root = tk.Tk()
root.title("Grid Example")

# Creating labels
label1 = tk.Label(root, text="Label 1", bg="red", fg="white")
label2 = tk.Label(root, text="Label 2", bg="green", fg="white")
label3 = tk.Label(root, text="Label 3", bg="blue", fg="white")

# Placing labels using grid
label1.grid(row=0, column=0, padx=10, pady=10, sticky="W")
label2.grid(row=1, column=1, padx=10, pady=10, sticky="E")
label3.grid(row=2, column=0, columnspan=2, padx=10, pady=10, sticky="EW")

# Creating and placing an Entry widget
entry = tk.Entry(root)
entry.grid(row=0, column=1, padx=10, pady=10, sticky="EW")

# Configuring the columns to expand and fill the space
root.columnconfigure(0, weight=1)
root.columnconfigure(1, weight=1)
```

root.mainloop()

In this example, three `Label` widgets and one `Entry` widget are placed in a window using the `grid` method. The `Label` widgets have different background (`bg`) and foreground (`fg`) colors for distinction. The `padx` and `pady` options add padding around the widgets. The `sticky` option is used to align the widgets within their grid cells. The `columnspan` option for `label3` makes it span across two columns. The `columnconfigure` method with the `weight` option is used to make the columns expand and fill the space when the window is resized, ensuring that the widgets adjust their sizes accordingly.

This example demonstrates how the `grid` geometry manager can be used to create a structured layout in a Tkinter application.