local themes = { horizontal = { theme = "horizontal", border = true, prompt_title = false, – results_title = false, sorting_strategy = "ascending", layout_strategy = "horizontal", }, }

local function get_layout(layout, opts) opts = opts or {} local theme_opts = themes[layout] return vim.tbl_deep_extend("force", theme_opts, opts) end

local function use_layout(picker, layout) return function() local lay = get_layout(layout) picker(lay) end end

return { "nvim-telescope/telescope.nvim", branch = "0.1.x", dependencies = { "nvim-lua/plenary.nvim", { "nvim-telescope/telescope-fzf-native.nvim", build = "make" }, "folke/todo-comments.nvim", "nvim-telescope/telescope-file-browser.nvim", "jvgrootveld/telescope-zoxide", },

config = function() local telescope = require("telescope") local actions = require("telescope.actions") local transform_mod = require("telescope.actions.mt").transform_mod

```
  local fb_actions = require("telescope").extensions.file_browser.actions
  local z_utils = require("telescope._extensions.zoxide.utils")
  local trouble = require("trouble")
  local trouble_telescope = require("trouble.sources.telescope")

  local custom_actions = transform_mod({
     open_trouble_qflist = function(prompt_bufnr)
        trouble.toggle("quickfix")
     end,
  })

  telescope.setup({
     defaults = {
        path_display = { "smart" },
        color_devicons = false, -- colored icons
        mappings = {
           i = {
              ["<C-k>"] = actions.move_selection_previous,
              ["<C-j>"] = actions.move_selection_next,

              -- preview scrolling
              ["<C-i>"] = actions.preview_scrolling_up,
              ["<C-u>"] = actions.preview_scrolling_down,

              -- select multiple
              ["<C-a>"] = actions.toggle_all,
              ["<C-BS>"] = actions.toggle_selection,
              ["<Tab>"] = actions.toggle_selection + actions.move_selection_worse, -- doesn
              ["<S-Tab>"] = actions.toggle_selection + actions.move_selection_better,
            ["<C-o>"] = actions.select_tab,
            ["<C-p>"] = require("telescope.actions.layout").toggle_preview,
            ["<c-d>"] = { "<C-u>", type = "command" },

            ["<C-q>"] = actions.send_selected_to_qflist + custom_actions.open_trouble_qfli
            ["<C-t>"] = trouble_telescope.open,
            ["<Esc>"] = actions.close,
         },
         -- n = {
         --    ["<Tab>"] = actions.toggle_selection + actions.move_selection_worse,
         --    ["<S-Tab>"] = actions.toggle_selection + actions.move_selection_better,
         --    ["<C-a>"] = actions.toggle_all,
         --    ["<C-BS>"] = actions.toggle_selection,
         -- },
      },

      layout_strategy = "vertical",
      layout_config = {
         vertical = {
            preview_cutoff = -1,
            height = 0.9,
            prompt_position = "bottom",
            width = 0.8,
         },
      },
   },

   pickers = {
      live_grep = {
         -- disable_devicons = true, -- disable file icons
         file_ignore_patterns = { "node_modules/", ".git/" },
         additional_args = function(_)
            return { "--hidden" } -- include hidden files
         end,
      },

      find_files = {
         -- disable_devicons = true, -- disable file icons
         file_ignore_patterns = { "node_modules/", ".git/" },
         find_command = { "fd", "--type", "f", "--hidden", "--exclude", ".DS_Store" }, --
      },

      oldfiles = {
         -- disable_devicons = true, -- disable file icons
         file_ignore_patterns = { "node_modules/", ".git/" },
      },
    grep_string = {
      -- disable_devicons = true, -- disable file icons
      file_ignore_patterns = { "node_modules/", ".git/" },
    },

    buffers = {
      mappings = {
        i = {
          ["<C-d>"] = function(prompt_bufnr) -- custom mapping to delete buffer
            local action_state = require("telescope.actions.state")
            local current_picker = action_state.get_current_picker(prompt_bufnr)
            local selected_entry = action_state.get_selected_entry()
            local bufnr_to_delete = selected_entry.bufnr
            vim.api.nvim_buf_delete(bufnr_to_delete, { force = true })
            current_picker:delete_selection(function(selection)
              return selection.bufnr == bufnr_to_delete
            end)
          end,
        },

        n = {
          ["<C-d>"] = function(prompt_bufnr) -- custom mapping to delete buffer
            local action_state = require("telescope.actions.state")
            local current_picker = action_state.get_current_picker(prompt_bufnr)
            local selected_entry = action_state.get_selected_entry()
            local bufnr_to_delete = selected_entry.bufnr
            vim.api.nvim_buf_delete(bufnr_to_delete, { force = true })
            current_picker:delete_selection(function(selection)
              return selection.bufnr == bufnr_to_delete
            end)
          end,
        },
      },
    },
  },

  extensions = {
    file_browser = {
      -- disable_devicons = true, -- disable file icons
      file_ignore_patterns = { "node_modules/", ".git/" },
      hidden = { file_browser = false, folder_browser = false },
      prompt_path = true,
      mappings = {
        i = {
          ["<C-h>"] = fb_actions.goto_home_dir,
                    ["<C-S-h>"] = fb_actions.toggle_hidden, -- toggle hidden files in file brow
                },
            },
        },

        zoxide = {
            prompt_title = "Zoxide Paths",
            mappings = {
                default = {
                    after_action = function(selection)
                        print("Update to (" .. selection.z_score .. ") " .. selection.path)
                    end,
                },
                ["<C-s>"] = {
                    before_action = function(selection)
                        print("before C-s")
                    end,
                    action = function(selection)
                        vim.cmd.edit(selection.path)
                    end,
                },
                -- Opens the selected entry in a new split
                ["<C-q>"] = { action = z_utils.create_basic_command("split") },
            },
        },

        fzf = {
            fuzzy = false,
            override_generic_sorter = true,
            override_file_sorter = true,
            case_mode = "smart_case",
        },
    },
})

telescope.load_extension("fzf")
telescope.load_extension("file_browser")
telescope.load_extension("zoxide")

local telescope_builtin = require("telescope.builtin")

-- set keymaps
local key = vim.keymap

key.set(
    "n",
    "<leader>fc",
    use_layout(telescope_builtin.current_buffer_fuzzy_find, "horizontal"),
    { desc = "Find in current buffer" }
)
key.set("n", "<leader>ff", "<cmd>Telescope find_files<cr>", { desc = "Find files in cwd" }
key.set("n", "<leader>fr", "<cmd>Telescope oldfiles<cr>", { desc = "Find recent files" })
key.set("n", "<leader>fs", "<cmd>Telescope live_grep<cr>", { desc = "Find string in cwd" }
key.set("n", "<leader>fg", "<cmd>Telescope grep_string<cr>", { desc = "Find string under c
key.set("n", "<leader>fb", "<cmd>Telescope buffers<cr>", { desc = "Show open buffers" })

key.set("n", "<leader>fh", "<cmd>TodoTelescope<cr>", { desc = "Find todos" })
key.set("n", "<space>ft", ":Telescope file_browser<CR>", { desc = "File browser" })
key.set("n", "<leader>fz", require("telescope").extensions.zoxide.list, { desc = "Zoxide F

vim.api.nvim_create_user_command("GrepOpenFiles", function()
    require("telescope.builtin").live_grep({ grep_open_files = true })
end, {}) -- search with telecope in opened files

key.set(
    "n",
    "<leader>fv",
    "<cmd>lua require('telescope.builtin').live_grep({search_dirs={vim.fn.expand('%:p')}})<
    { desc = "Find string in current file" }
)
```

end, }