In main.go, you have this line in the config struct:

DJIAPIKey string `env:"DJI\_API\_KEY,required"`

This line is doing a few important things:

1. It's declaring a field named DJIAPIKey of type string in the config struct.
2. The env:"DJI\_API\_KEY,required" tag is used by the environment variable parsing library (in this case, github.com/caarlos0/env/v10 as seen in your imports).
3. This tag tells the parsing library to:
   * Look for an environment variable named DJI\_API\_KEY
   * Populate the DJIAPIKey field with the value of that environment variable
   * Treat this variable as required (the program will error if it's not set)

In your main() function, you have this code:

cfg := config{}

if err := env.Parse(&cfg); err != nil {

panic(fmt.Sprintf("failed to parse env config: %v", err))

}

This is where the magic happens:

1. It creates an empty config struct.
2. It calls env.Parse(&cfg), which reads environment variables and populates the cfg struct based on the env tags.
3. After this call, cfg.DJIAPIKey will contain the value of the DJI\_API\_KEY environment variable.

So, when you use cfg.DJIAPIKey in your code, you're accessing the value that was read from the environment variable. The manifest.yaml file ensures that this environment variable is set when your application runs.

You're correct that you can reference it in main.go with cfg.DJIAPIKey. This is not writing it as an environment variable, but rather reading the value that was set by your container orchestration system (based on the manifest.yaml file) and populated into your config struct by the env.Parse() function.

In parse\_txt.go, the NewTXTParser() function is directly reading the environment variable:

func NewTXTParser() \*TXTParser {

return &TXTParser{

apiKey: os.Getenv("DJI\_API\_KEY"),

}

}

This is another way to access the same environment variable, but it's done directly rather than through the config struct.

Both methods are reading the same environment variable, just in different ways and at different points in your application's lifecycle.