Sistemas distribuidos:

La computación distribuida es un modelo para resolver problemas de computación masiva utilizando un gran número de ordenadores organizados en clústeres incrustados en una infraestructura de telecomunicaciones distribuida.
Julia no le impone al usuario ningún estilo de paralelismo en particular. En vez de esto, le provee con bloques de construcción clave para la computación distribuida, logrando hacer lo suficientemente flexible el soporte de varios estilos de paralelismo y permitiendo que los usuarios añadan más. El siguiente ejemplo demuestra de manera simple como contar el número de caras de una gran cantidad de volados en paralelo.

```
julia> using Distributed
julia> nheads = @distributed (+) for i = 1:200000000
           Int(rand(Bool))
           end
99998199
```

Por ejemplo, los clústeres de Beowulf se admiten a través de un administrador de clústeres personalizado implementado en el paquete ClusterManagers.jl.

```
addprocs(manager::ClusterManager; kwargs...) -> List of process identifiers
```

Inicia procesos de trabajo a través del administrador de clúster especificado.

Variables de entorno:
Si el proceso no logra establecer una conexión con un worker recién iniciado en 60.0 segundos, el worker lo trata como una situación fatal y termina. Este tiempo de espera se puede controlar mediante la variable de entorno JULIA_WORKER_TIMEOUT. El valor de JULIA_WORKER_TIMEOUT en el proceso maestro especifica la cantidad de segundos que un trabajador recién lanzado espera para el establecimiento de la conexión.

A modo de curiosidad y para profundizar un poco más sobre este tema, adjuntamos un repositorio público llamado “La guía definitiva para los sistemas distribuidos en Julia”.
https://github.com/juliohm/julia-distributed-computing

Fuente y documentación oficial acerca de Sistemas Distribuidos:
https://docs.julialang.org/en/v1/stdlib/Distributed/

Dockers:

Docker le. Docker es un sistema operativo para contenedores. De manera similar a cómo una máquina virtual virtualiza (elimina la necesidad de administrar directamente) el hardware del servidor, los contenedores virtualizan el sistema operativo de un servidor. Docker se instala en cada servidor y proporciona comandos sencillos que puede utilizar para crear, iniciar o detener contenedores. Proporciona una manera estándar de ejecutar su código.

Si bien esto es compatible con Julia, llegamos a la conclusión de que no es una utilidad muy aprovechada y tal vez no tenga tanta lógica ponerse a analizar profundamente esto.
Sin embargo, investigamos y encontramos un poco de trabajo y documentación al respecto.
Adjuntamos enlaces:
https://hub.docker.com/r/jupyter/datascience-notebook/dockerfile
https://sebastiancallh.github.io/post/julia-in-jupyter/