

# Refs



# Ref

Why: purely functional, thread-safe state management

Purely functional atomic reference

```
val atomicMoL = Ref.make(42)
```

Interacting with a Ref is an effect

- setting new value
- getting existing value
- getting + setting atomically
- updating with a function
- updating + getting (old value or new value)
- modifying + surfacing an external value

```
val modifiedMoL = atomicMoL.flatMap { ref =>  
  ref.modify(value => (s"my current goal is $value", value * 10))  
}
```

**ZIO rocks**

