🎯 Quick Reference Card

One-Command Starts

Test a Project

```
# Connection Pool (READY - has tests)
cd /home/zjunaidz/AI/go-concurrency/projects/connection-pool/final && go test -v

# Pub-Sub (READY - has tests)
cd /home/zjunaidz/AI/go-concurrency/projects/pub-sub/final && go test -v
```

Run Benchmarks

```
# From any project's final/ directory
go test -bench=. -benchmem

# Or benchmark all projects
cd /home/zjunaidz/AI/go-concurrency && ./run_all_benchmarks.sh
```

Check for Race Conditions

```
# Always use -race flag in development
go test -race -v
```

Common Commands

Task	Command
Run all tests	`cd /home/zjunaidz/AI/go-concurrency && ./run_all_tests.sh`
Setup projects	`cd /home/zjunaidz/AI/go-concurrency && ./setup.sh`
Run specific test	`go test -run=TestName -v`
Run benchmarks	`go test -bench=. -benchmem`
Check race conditions	`go test -race -v`
Build project	`go build .`
Format code	`go fmt ./...`
Update dependencies	`go mod tidy`

Project Status

Project	Location	Tests	Status

connection-pool	`projects/connection-pool/final/`	✅	**USE THIS**
pub-sub	`projects/pub-sub/final/`	✅	**USE THIS**
rate-limiter	`projects/rate-limiter/final/`	⚠️	Code only
job-queue	`projects/job-queue/final/`	⚠️	Code only
cache	`projects/cache/final/`	⚠️	Code only
web-crawler	`projects/web-crawler/final/`	⚠️	File needs fix

Key Patterns Cheat Sheet

Worker Pool

```
// Start workers
for i := 0; i < numWorkers; i++ {
    wg.Add(1)
    go worker(ctx, jobs)
}

// Worker function
func worker(ctx context.Context, jobs <-chan Job) {
    defer wg.Done()
    for {
        select {
        case job := <-jobs:
            process(job)
        case <-ctx.Done():
            return
        }
    }
}
```

Graceful Shutdown

```
// Setup
ctx, cancel := context.WithCancel(context.Background())
var wg sync.WaitGroup

// Shutdown
cancel()               // Signal cancellation
wg.Wait()              // Wait for goroutines
close(channels)        // Close channels
```

Rate Limiting

```
// Token bucket
ticker := time.NewTicker(rate)
defer ticker.Stop()

<-ticker.C  // Wait for token
doWork()    // Process
```

Circuit Breaker

```
// Track failures
failures++
if failures >= threshold {
    state = open
    lastFailure = time.Now()
}

// Check state
if state == open {
    if time.Since(lastFailure) > timeout {
        state = halfOpen
    } else {
        return ErrCircuitOpen
    }
}
```

Sharding

```
// Hash to shard
index := hash(key) % numShards
shard := shards[index]

shard.mu.Lock()
defer shard.mu.Unlock()
// Access shard data
```

Test Flags

Flag	Purpose	Example
`-v`	Verbose output	`go test -v`
`-race`	Race detector	`go test -race`
`-bench=.`	Run benchmarks	`go test -bench=.`
`-benchmem`	Memory stats	`go test -bench=. -benchmem`
`-run=Name`	Run specific test	`go test -run=TestPool`

`-count=N`	Run N times	`go test -count=10`
`-parallel=N`	N parallel tests	`go test -parallel=4`
`-timeout=T`	Test timeout	`go test -timeout=30s`
`-cpuprofile=f`	CPU profile	`go test -cpuprofile=cpu.prof`
`-memprofile=f`	Memory profile	`go test -memprofile=mem.prof`

Debugging

Find Deadlocks

```
# Send SIGQUIT to running program
kill -QUIT <pid>

# Or use CTRL+\ in terminal
# Shows all goroutine stacks
```

Profile Performance

```
# CPU profiling
go test -cpuprofile=cpu.prof -bench=.
go tool pprof cpu.prof

# Memory profiling
go test -memprofile=mem.prof -bench=.
go tool pprof mem.prof
```

Race Detector

```
# Always test with race detector
go test -race ./...

# Build with race detector
go build -race .

# Run with race detector
go run -race main.go
```

Performance Targets

Project	Metric	Target
rate-limiter	ops/sec	500k+ (with sharding)
job-queue	jobs/sec	10k+

cache	ops/sec	100M+ (256 shards)
web-crawler	pages/min	200-500 (polite)
connection-pool	ops/sec	50k+
pub-sub	msgs/sec	100k+

Documentation Map

```
/home/zjunaidz/AI/go-concurrency/
├── START_HERE.md          ← Read this first!
├── GETTING_STARTED.md     ← Comprehensive guide
├── PROGRESS.md            ← Learning tracks
├── README.md              ← Overview
├── setup.sh               ← Initialize all projects
├── run_all_tests.sh       ← Test everything
├── run_all_benchmarks.sh  ← Benchmark everything
│
├── 00-foundations/        ← Start here for theory
├── 01-go-concurrency-primitives/
├── 02-memory-model/
├── 03-classic-problems/
├── 04-patterns/
├── 05-real-world-go/
├── 06-testing-and-debugging/
├── 07-lld-hld/
├── 08-interview-prep/     ← 50+ interview Q&A
│
└── projects/              ← Hands-on practice
    ├── connection-pool/   ✅ Tests ready
    ├── pub-sub/           ✅ Tests ready
    ├── rate-limiter/      ⚠️  Code only
    ├── job-queue/         ⚠️  Code only
    ├── cache/             ⚠️  Code only
    └── web-crawler/       ⚠️  Needs fix
```

Learning Progression

1. **Foundations** (1-2 days)
   - Read: `00-foundations/` and `01-go-concurrency-primitives/`
   - Practice: Run connection-pool tests
2. **Patterns** (3-5 days)
   - Read: `04-patterns/` and `05-real-world-go/`
   - Practice: Study all 6 projects
3. **Production** (1-2 weeks)
   - Read: `06-testing-and-debugging/`

- Practice: Modify projects, add features

4. **Interviews** (1 week)
   - Read: `08-interview-prep/`
   - Practice: Whiteboard project designs

Quick Fixes

Problem: "go: no module"

```
cd /home/zjunaidz/AI/go-concurrency
./setup.sh
```

Problem: "cannot find package"

```
cd project/final
go mod tidy
```

Problem: "too many open files"

```
ulimit -n 4096
```

Problem: Tests timeout

```
go test -timeout=60s -v
```

Useful Aliases

Add to your `~/.bashrc` or `~/.zshrc`:

```
# Go concurrency shortcuts
alias goc='cd /home/zjunaidz/AI/go-concurrency'
alias gop='cd /home/zjunaidz/AI/go-concurrency/projects'
alias gotr='go test -race -v'
alias gotb='go test -bench=. -benchmem'
```

Then reload: `source ~/.bashrc`

Interview Prep Checklist

- [ ] Can explain goroutine vs thread
- [ ] Know when to use buffered vs unbuffered channels
- [ ] Understand mutex vs RWMutex vs atomic
- [ ] Can implement worker pool from memory
- [ ] Know how to prevent deadlocks

- [ ] Can explain memory model happens-before
- [ ] Comfortable with context cancellation
- [ ] Can design graceful shutdown
- [ ] Understand sharding for lock reduction
- [ ] Know circuit breaker pattern

Resources

- **Start**: START_HERE.md
- **Guide**: GETTING_STARTED.md
- **Track**: PROGRESS.md
- **Docs**: `00-foundations/` through `08-interview-prep/`

First Command:

```
cd /home/zjunaidz/AI/go-concurrency/projects/connection-pool/final && go test -v
```