An Introduction to PocketBase:

A Go-Based Backend as a Service

by Haseeb Majid

About Me

- Haseeb Majid
  - Backend Software Engineer at Curve
  - https://haseebmajid.dev
- Loves cats 🐱
- Avid cricketer 🏏 #BazBall

Adding a new side project to the list of unfinished side projects

What is a Backend as a Service (BaaS)?

Handle the basic repetitive tasks

Popular BaaS

- Firebase
- Supabase
- Amplify

Why use PocketBase?

- Runs from a single binary
- Written in Go
  - Extend as framework
- Easy to use Dashboard

Dashboard

Use as a Framework

```
go mod init gitlab.com/hmajid2301/talks/.../example
go get github.com/pocketbase/pocketbase
```

```go
// main.go

package main

import (
    "log"

    "github.com/pocketbase/pocketbase"
)

func main() {
    app := pocketbase.New()

    if err := app.Start(); err != nil {
        log.Fatal(err)
    }
```

```go
package main

import (
	"log"

	"github.com/pocketbase/pocketbase"
)

func main() {
	app := pocketbase.New()

	if err := app.Start(); err != nil {
		log.Fatal(err)
	}
}
```

```
go run main.go serve --http=localhost:8080
```

Add a Route

```go
# main.go

import (
	"net/http"

	"github.com/labstack/echo/v5"
	"github.com/pocketbase/pocketbase"
	"github.com/pocketbase/pocketbase/apis"
	"github.com/pocketbase/pocketbase/core"
)

func main() {
	//...

	app.OnBeforeServe().Add(func(e *core.ServeEvent) error {
		e.Router.POST("/comment", handler, middlewares)
```

```go
app.OnBeforeServe().Add(func(e *core.ServeEvent) error {
    e.Router.POST("/comment",

      //handler
      func(c echo.Context) error {
          return c.NoContent(http.StatusCreated)
      },

      //middlewares
      apis.ActivityLogger(app),
      apis.RequireRecordAuth(),
    )
    return nil
})
```

Client Code

```
import PocketBase from "pocketbase";

const pb = new PocketBase("http://127.0.0.1:8080");
// code to auth user
// ...

await pb.send("/comment", {
  // for all possible options check
  // https://developer.mozilla.org/en-US/docs/Web/API/fetch#options
});
```

Add Record to DB

```go
// ...
type Comments struct {
    models.BaseModel
    Post    string `db:"post" json:"post"`
    User    string `db:"user" json:"user"`
    Message string `db:"message" json:"message"`
}

func (c *Comments) TableName() string {
    return "comments"
}

var _ models.Model = (*Comments)(nil)
func main() {
  // ...
```

Add Record to DB

```go
app.OnBeforeServe().Add(func(e *core.ServeEvent) error {
e.Router.POST("/comment",
// handler
func(c echo.Context) error {
  auth, _ := c.Get(apis.ContextAuthRecordKey).(*models.Recor
  m := fmt.Sprintf("Hi 👋 from %s", auth.Username())
  commentRecord := &Comments{
    User:    auth.Id,
    Message: m,
    Post: "1",
  }

  err := app.Dao().Save(commentRecord)
  if err != nil {
    return err
  }
```

Add Record to DB

```go
      commentRecord := &Comments{
        User:    auth.Id,
        Message: m,
        Post: "1",
      }

      err := app.Dao().Save(commentRecord)
      if err != nil {
        return err
      }

      return c.NoContent(http.StatusCreated)
    },

    // middlewares
    apis.ActivityLogger(app),
```

Migrations

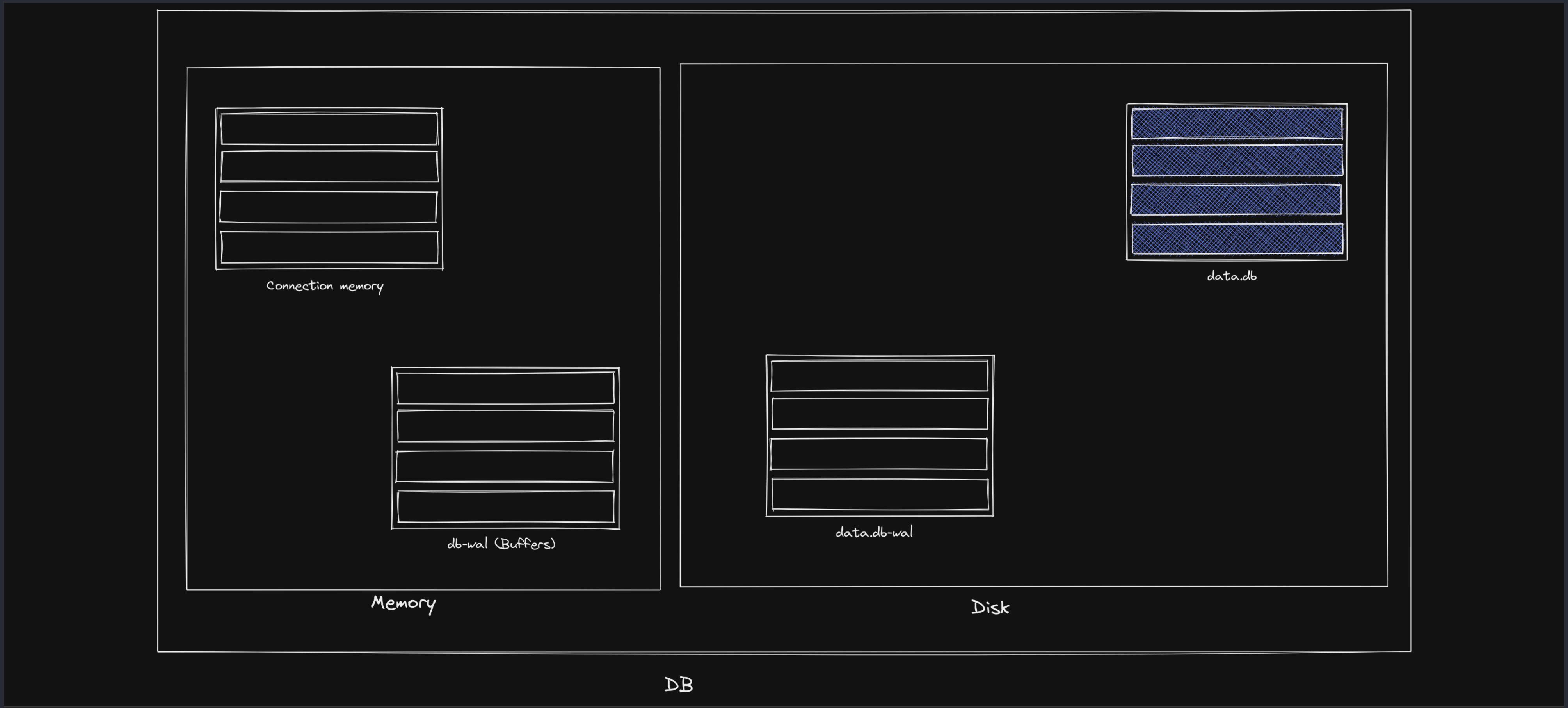
```
ls -al migrations/
Permissions  User   Group   Date Modified Name
.rw-r--r--   haseeb haseeb  2 Apr 22:52  1680445294_created_posts.go
.rw-r--r--   haseeb haseeb  2 Apr 22:52  1680445383_created_comments
.rw-r--r--   haseeb haseeb  2 Apr 22:52  1680445466_updated_comments
.rw-r--r--   haseeb haseeb  2 Apr 22:52  1680445481_updated_posts.go
```

```go
// main.go
package main

import (
    "log"

    "github.com/pocketbase/pocketbase"
    "github.com/pocketbase/pocketbase/plugins/migratecmd"

    // you must have have at least one
    // .go migration file in the "migrations" directory
    _ "gitlab.com/hmajid2301/talks/.../migrations"
)

func main() {
  app := pocketbase.New()
)

func main() {
  app := pocketbase.New()

  migratecmd.MustRegister(
    app,
    app.RootCmd,
    &migratecmd.Options{
      // auto creates migration files
      // when making collection changes
      Automigrate: true,
  })

  // ...
}
```

SQLite

- Does it Scale?
    - Write-Ahead Logging (WAL mode)

What is WAL Mode?



Why use WAL Mode?

- Is significantly faster in most scenarios.
- WAL uses many fewer `fsync()` operations
- Provides more concurrency as a writer does not block readers.

Testing

```go
package main

import (
    "net/http"
    "testing"

    "github.com/pocketbase/pocketbase/tests"
    "github.com/pocketbase/pocketbase/tokens"
)

// username: test@example.com
// password: password11
const testDataDir = "./tests/pb_data"

func TestCommentEndpoint(t *testing.T) {
    recordToken, err := generateRecordToken("users", "test@exampl
```

Testing

```go
      ExpectedEvents:  map[string]int{"OnModelAfterCreate": 1,
                                      "OnModelBeforeCreate": 1},
      TestAppFactory:  setupTestApp,
    },
  }

  for _, scenario := range scenarios {
    scenario.Test(t)
  }
}

func generateRecordToken(collectionNameOrId string, email string)
    app, err := tests.NewTestApp(testDataDir)
    if err != nil {
        return "", err
```

Testing

```go
    "testing"

    "github.com/pocketbase/pocketbase/tests"
    "github.com/pocketbase/pocketbase/tokens"
)

// username: test@example.com
// password: password11
const testDataDir = "./tests/pb_data"

func TestCommentEndpoint(t *testing.T) {
    recordToken, err := generateRecordToken("users", "test@exampl
    if err != nil {
        t.Fatal(err)
    }
```

Deploy

Dockerfile

```dockerfile
FROM golang:1.20-alpine as builder

WORKDIR /build
RUN apk update && apk upgrade && \
    apk add --no-cache ca-certificates && \
    update-ca-certificates

COPY . .
RUN CGO_ENABLED=0 GOOS=linux go build -o app main.go

FROM scratch
COPY --from=builder /build/app .
COPY --from=builder /etc/ssl/certs/ca-certificates.crt \
                    /etc/ssl/certs/

ENTRYPOINT [ "./app" ]
```

Dockerfile

```
WORKDIR /build
RUN apk update && apk upgrade && \
    apk add --no-cache ca-certificates && \
    update-ca-certificates

COPY . .
RUN CGO_ENABLED=0 GOOS=linux go build -o app main.go

FROM scratch
COPY --from=builder /build/app .
COPY --from=builder /etc/ssl/certs/ca-certificates.crt \
                    /etc/ssl/certs/

ENTRYPOINT [ "./app" ]
CMD ["serve", "--http=0.0.0.0:8080"]
```

fly.io

```toml
# fly.toml
app = "example"
kill_signal = "SIGINT"
kill_timeout = 5
processes = []

[build]
dockerfile = "Dockerfile"

[env]
ENV = "production"

[experimental]
allowed_public_ports = []
auto_rollback = true
enable_consul = true
```

fly.io

```toml
handlers = ["http"]
port = 80

[[services.ports]]
handlers = ["tls", "http"]
port = 443

[[services.tcp_checks]]
grace_period = "1s"
interval = "15s"
restart_limit = 0
timeout = "2s"

[mounts]
destination = "/pb_data"
source = "pb_data"
```

```
fly deploy
```

Gitlab CI

```yaml
deploy:
  stage: deploy
  only:
    - main
  image: docker
  services:
    - docker:dind
  before_script:
    - apk add curl
    - curl -L https://fly.io/install.sh | sh
  script:
    - fly deploy
```

Gitlab CI

```yaml
deploy:
  stage: deploy
  only:
    - main
  image: docker
  services:
    - docker:dind
  before_script:
    - apk add curl
    - curl -L https://fly.io/install.sh | sh
  script:
    - fly deploy
```

Other Features

- Expanding Relations
  - Join tables without making additional request
- Uploading Files
- API to manage (DB) backups

Caveats

- Need to self-host
  - PocketHost
- Does not have a stable API yet
- Can only scale vertically
  - LiteFS

Any Questions?

- Code: https://gitlab.com/hmajid2301/talks/an-intro-to-pocketbase
- Slides: https://haseebmajid.dev/talks/an-intro-to-pocketbase/

Useful Links

- PocketBase Docs
- Fireship Video on PocketBase
- WAL Mode Explained
- LiteFS
- My App Built Using PocketBase