

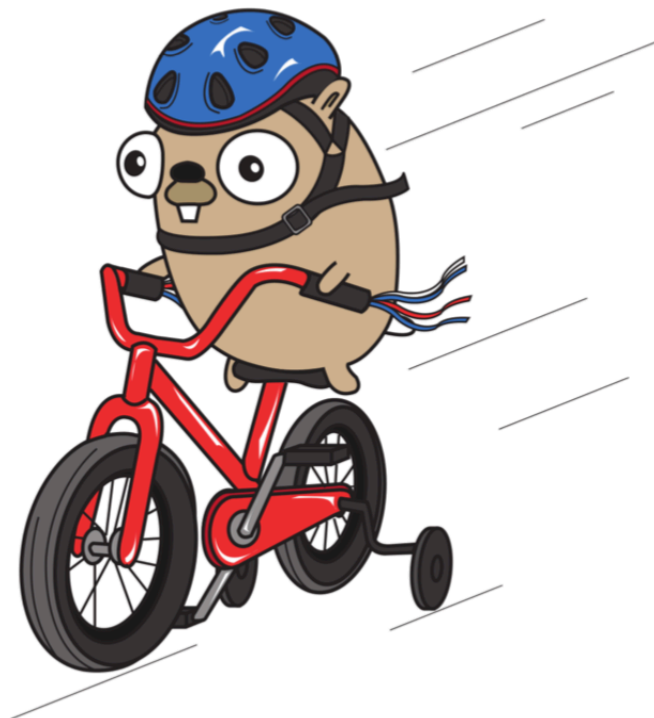
# git & golang book



**Read golang book and work with git**

# golang book

AN INTRODUCTION TO  
**PROGRAMMING**  
IN **GO**



CALEB DOXSEY

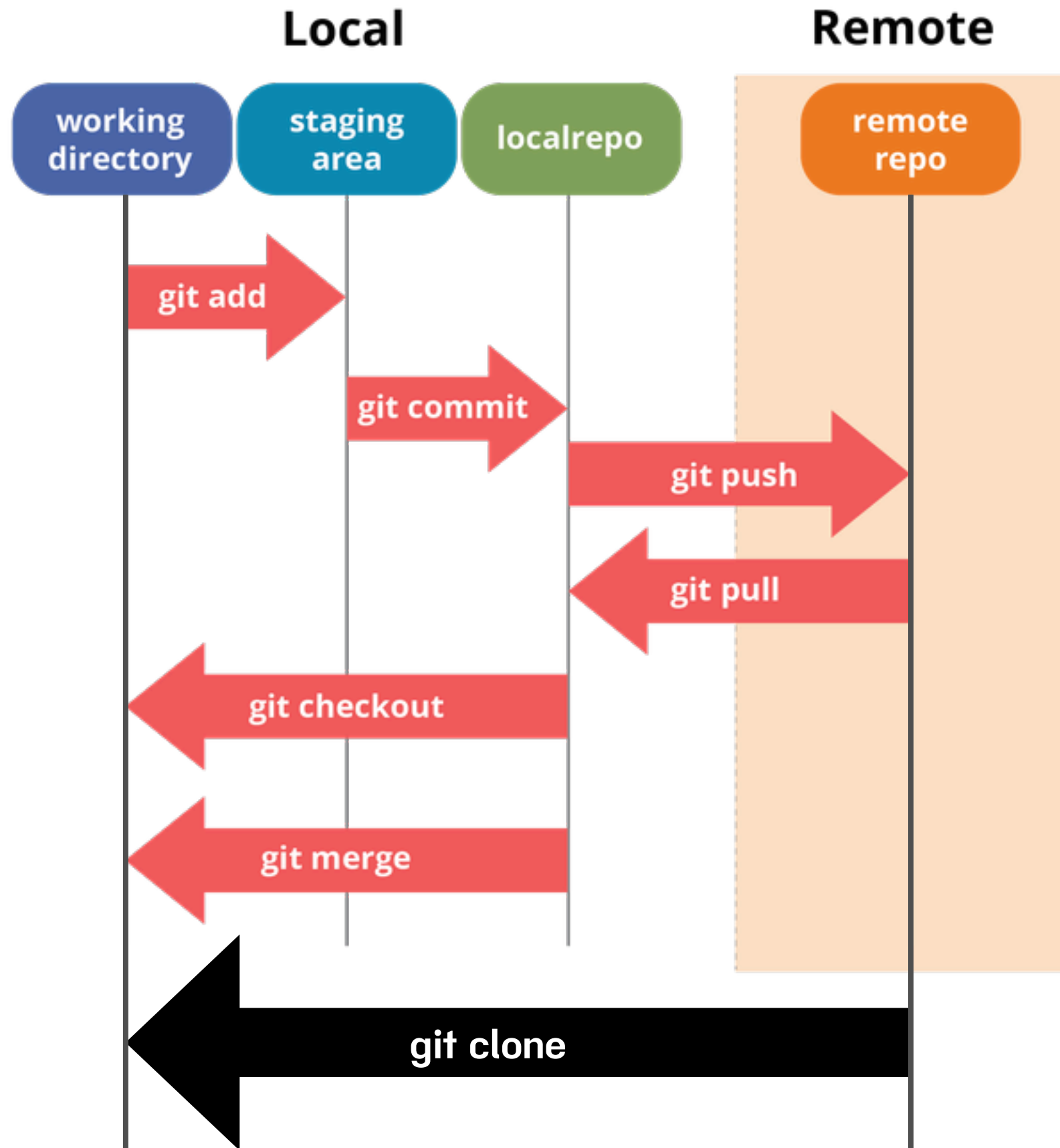
# git

**github account:**

`https://github.com/[yourname]`

example:

`https://github.com/boyone`



# Clone go-101

**clone go-101 to your workspace:**

```
>git clone https://github.com/boyone/go-101.git
```

# golang book [1]

**make working directory: [windows]**

```
>md src\dojo\golang-book
```

**make working directory: [linux/Mac]**

```
>mkdir -p src/dojo/golang-book
```

# golang book [2]

**go to golang-book directory:**

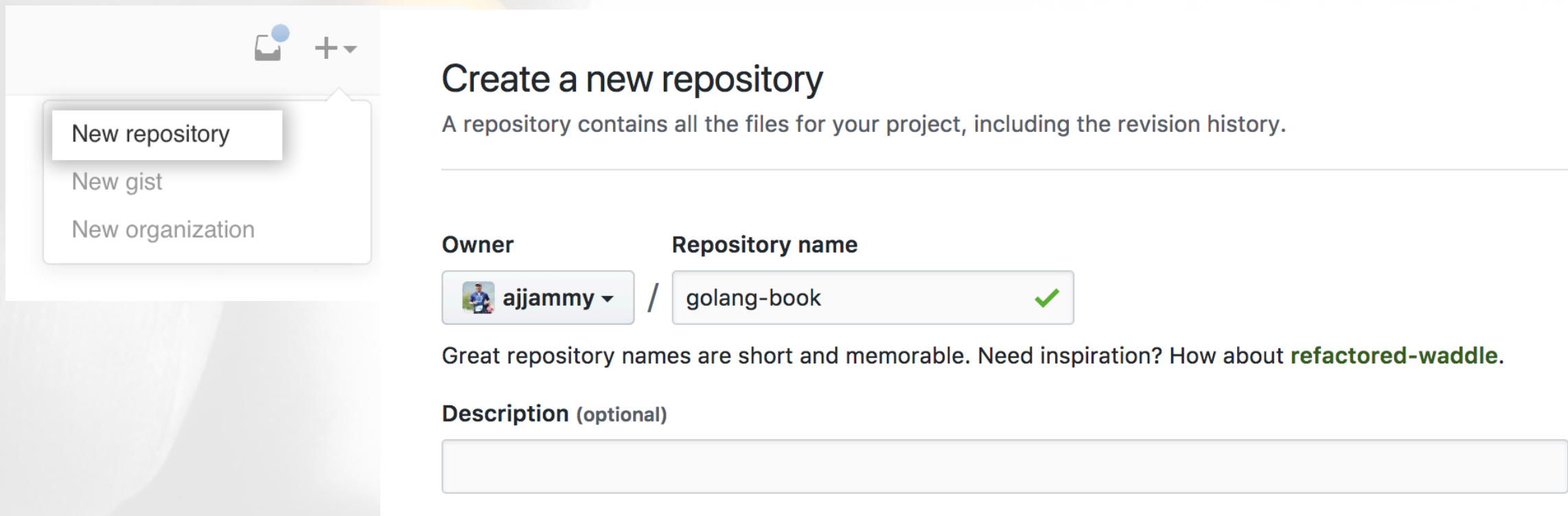
```
>cd src\dojo\golang-book
```

**git init for golang-book directory:**

```
>git init
```


# golang book [3]

## create github repository:



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner:  ajjammy

Repository name:  ✓

Great repository names are short and memorable. Need inspiration? How about [refactored-waddle](#).

Description (optional)

## add remote:

```
>git remote add origin https://github.com/<user>/golang-book.git
>git remote -v
```



# golang book [4]

create README.md file:

```
1  # Go Book
2
3  **Name:** *Chamnan Inta*
4
5  **Nickname:** *Jammy*
6
7  **Job Title:** *Programmer*
8
9  ## Chapter 2
10
11 ## Chapter 3
12
13 ## Chapter 4
```

# golang book [5]

**git add / git commit :**

```
>git add README.md
```

```
>git commit -m "Add README.md file"
```

**git push**

```
>git push -u origin master
```

# golang book [6]

create .gitignore file:

```
.gitignore x
1 *.exe
2 *.DS_Store
```

# golang book [7]

**git add / git commit :**

```
>git add .gitignore  
>git commit -m "Add .gitignore file"
```

**git push**

```
>git push
```

# golang book [8]

1. Read book chapter 2
2. Update README.md
3. Create main.go file at directory golang-book/chapter2-1

```
i README.md ●
1  # Go Book
2
3  **Name:** *Chamnan Inta*
4
5  **Nickname:** *Jammy*
6
7  **Job Title:** *Programmer*
8
9  ## Chapter 2
10
11  * chapter2-1 : My First Program
12
13  ## Chapter 3
```

```
🦉 main.go ×
1  package main
2
3  import "fmt"
4
5  // this is a comment
6  func main() {
7      fmt.Println("Hello World")
8  }
9
```



# golang book [9]


**git add / git commit :**

```
>git add .  
>git commit -m "Add chapter2-1 My first program"
```

**git push**

```
>git push
```

# add collaborators

 **ajjammy / golangbook**

Unwatch 3

★ Star 0

🍴 Fork 0

<> Code

! Issues 0

🔗 Pull requests 0

📁 Projects 0

📖 Wiki

📊 Insights

⚙️ Settings

Options

**Collaborators**

Branches


Webhooks


Integrations & services


Deploy keys


**Collaborators**

Push access to the repository

 **Thawatchai Jongsuwanpaisan**  
boyone



 ployploy




Search by username, full name or email address

You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

ajjammy|

Add collaborator

 **ajjammy** Jammy

# Exercise



Read book  
Chapter 1 to 4



Create folder chapter<...>-...  
Create file main.go  
Update README.md file

```
git add  
git commit  
git push
```



# golang : type Zero Value

```
package main

import "fmt"

func main() {
    fmt.Println("====Zero Value====")
    var number int
    var str string
    var boolean bool
    fmt.Printf("number: %v\n", number)
    fmt.Printf("str: '%v'\n", str)
    fmt.Printf("boolean: %v\n", boolean)
}
```

# golang : type Strings

```
package main

import "fmt"

func main() {
    fmt.Println("\n====String====")
    backticks := `hello world!,
today's good day.`
    fmt.Println(backticks)

    doubleQuotes := "hello world!,\ntoday's good day."
    fmt.Println(doubleQuotes)
}
```

# golang : type Floating point [1]

```
package main

import "fmt"

func main() {
    fmt.Println("====Floating point====")
    third := 1.0 / 3.0
    fmt.Printf("third = %v\n", third)
    fmt.Printf("third + third + third = %v\n", third+third+third)
}
```

# golang : type Floating point [2]

```
package main

import "fmt"

func main() {
    fmt.Println("====Comparing floating point====")
    fmt.Println("0.1 + 0.2 == 0.3 is", 0.1+0.2 == 0.3)
    num := 0.1
    num += 0.2
    fmt.Println("num == 0.3 is", num == 0.3)
    fmt.Println("num is", num)
}
```