Keeping Rails Applications on Track with Brakeman

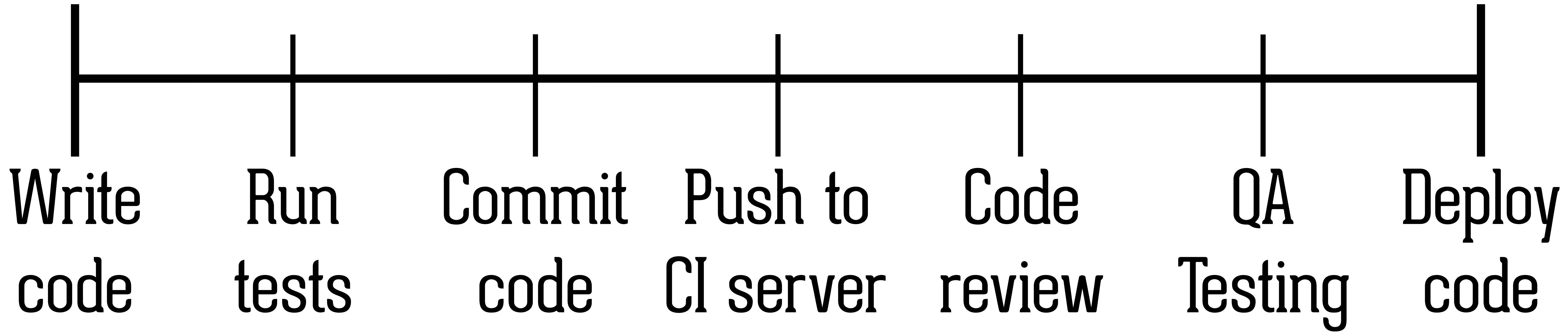
Justin Collins
@presidentbeef

RailsConf 2012

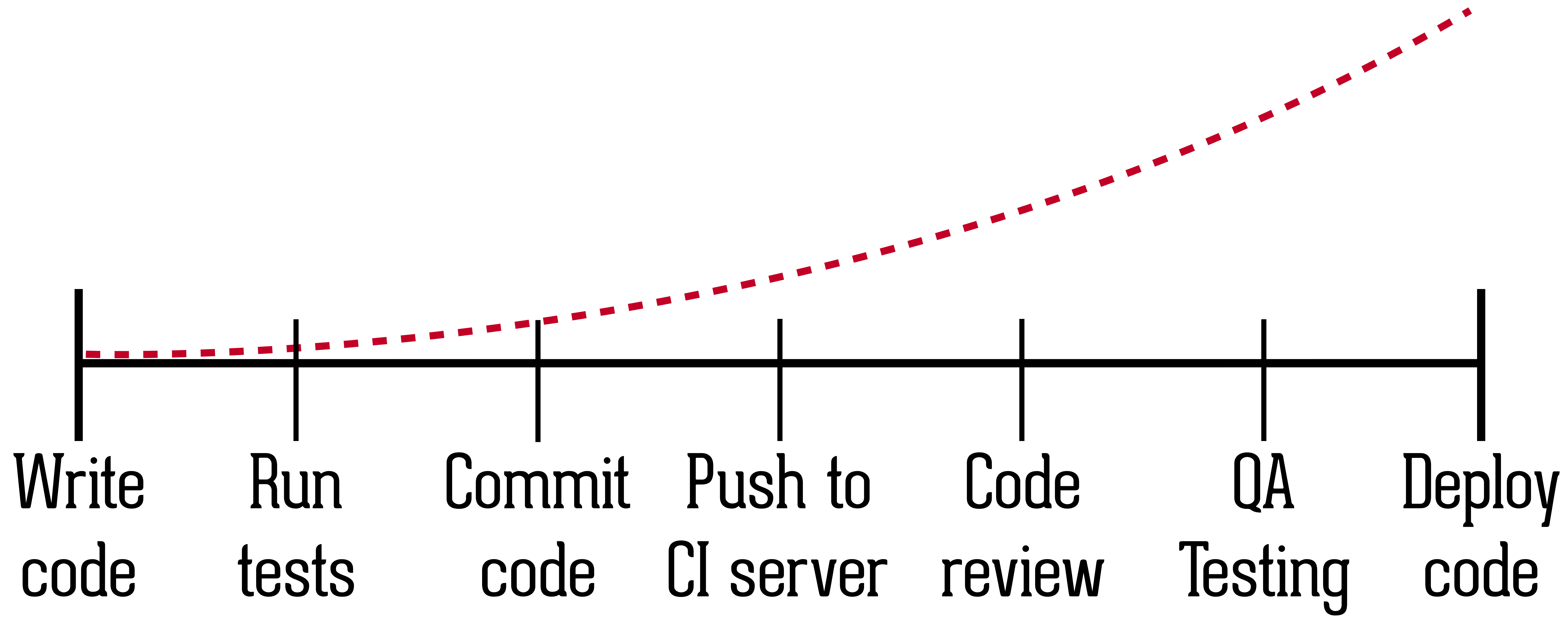
Everyone knows they “should” worry about security

But when should you worry?

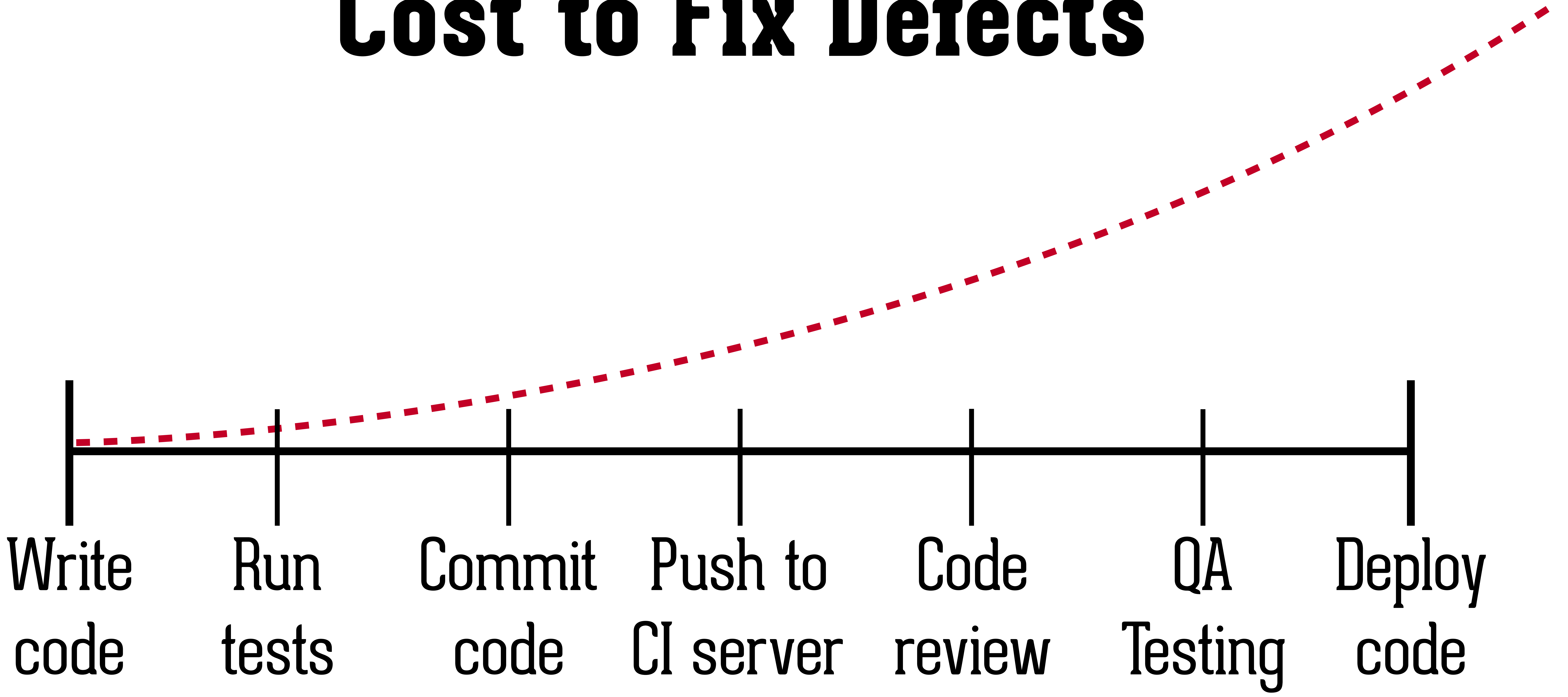
Idealized Software Development



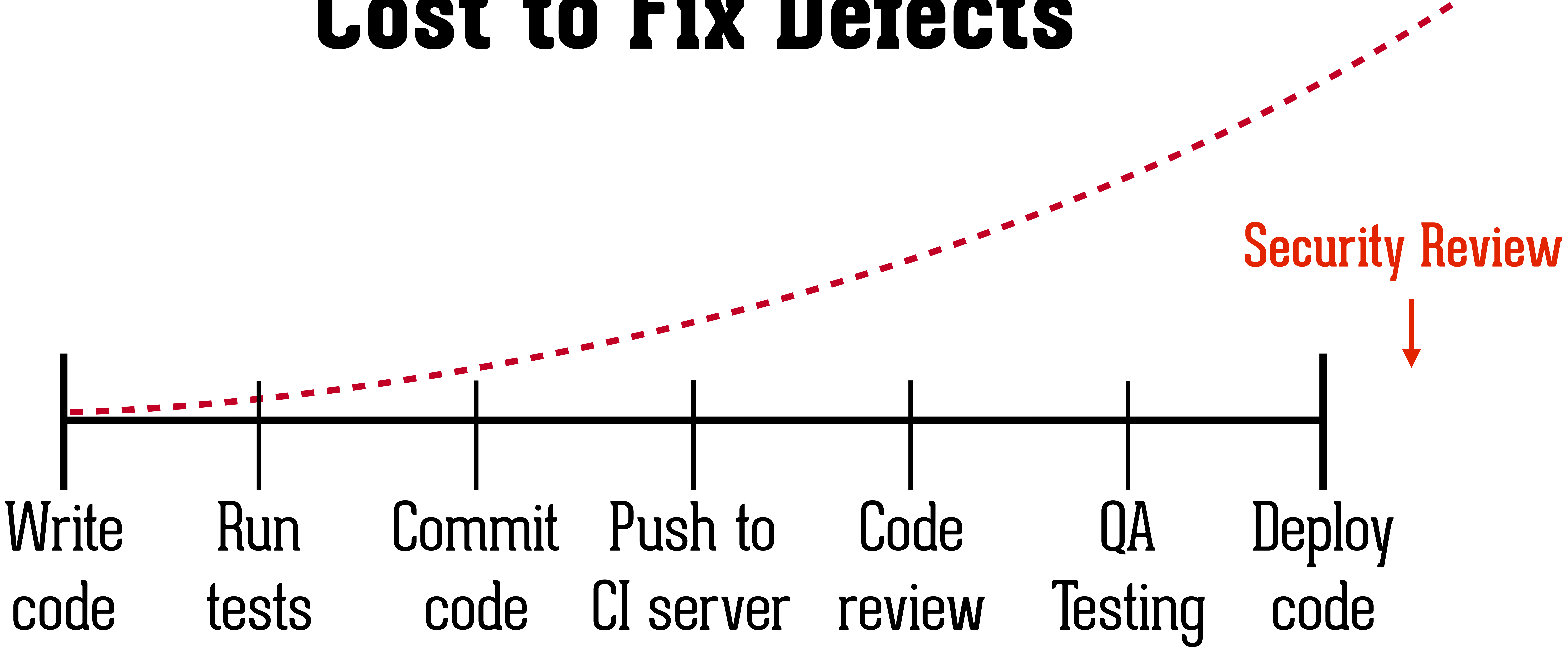
Cost to Fix Defects



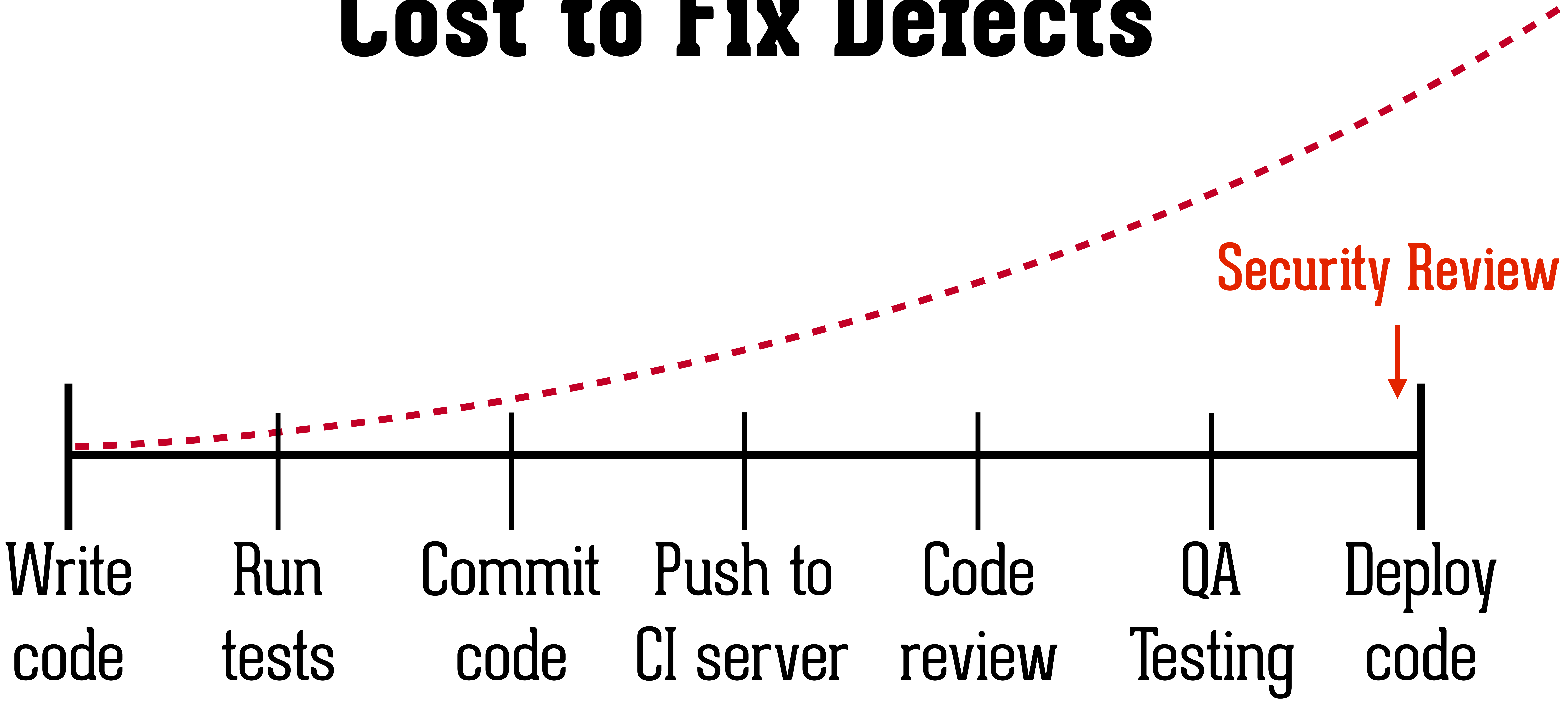
Cost to Fix Defects



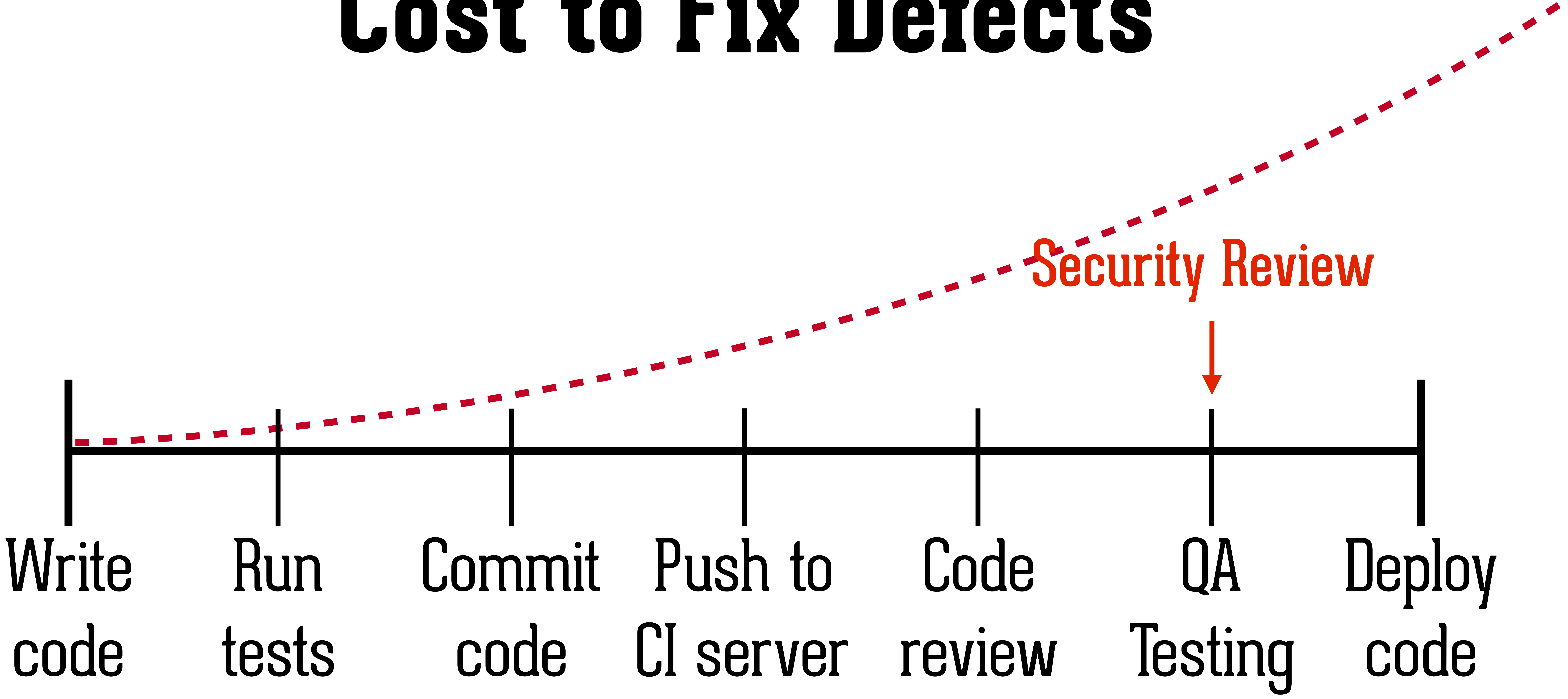
Cost to Fix Defects



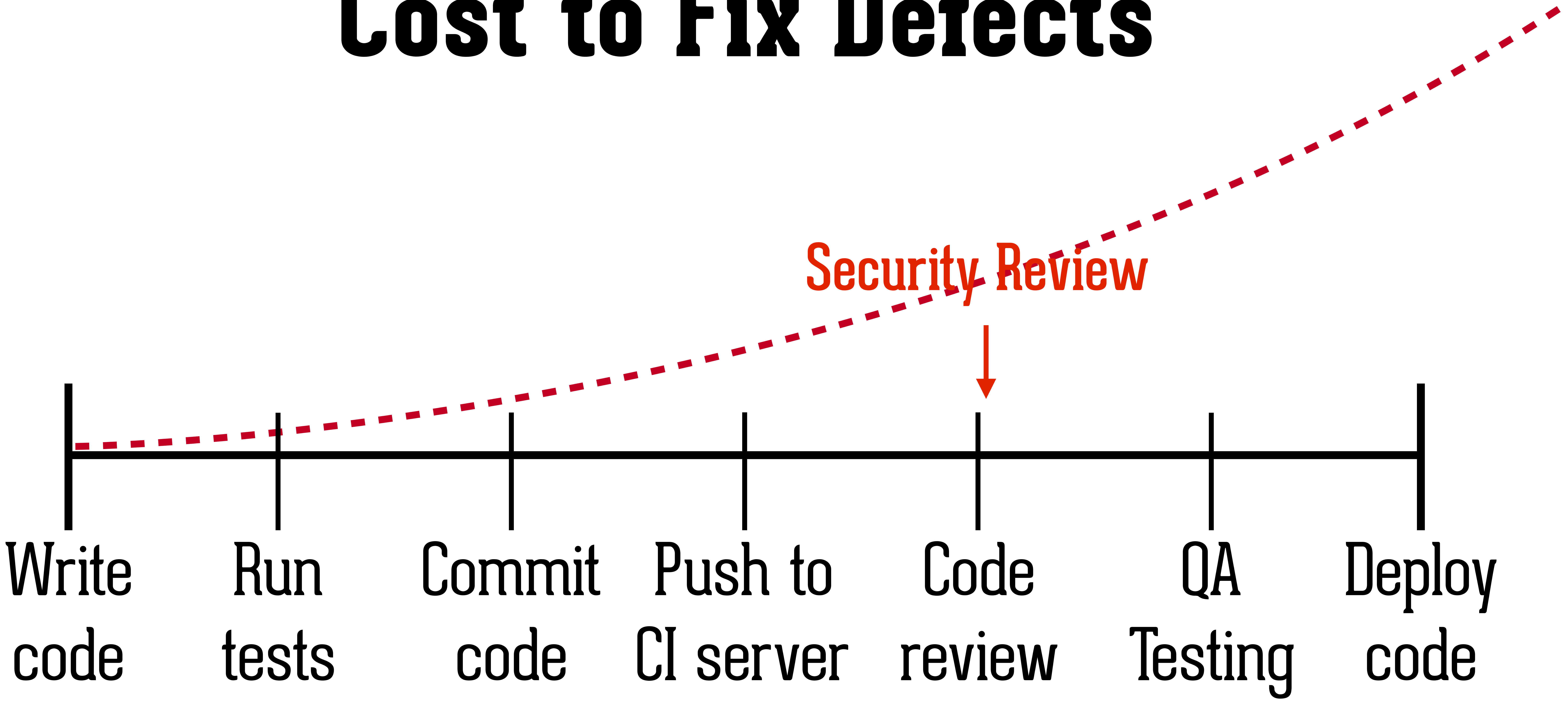
Cost to Fix Defects



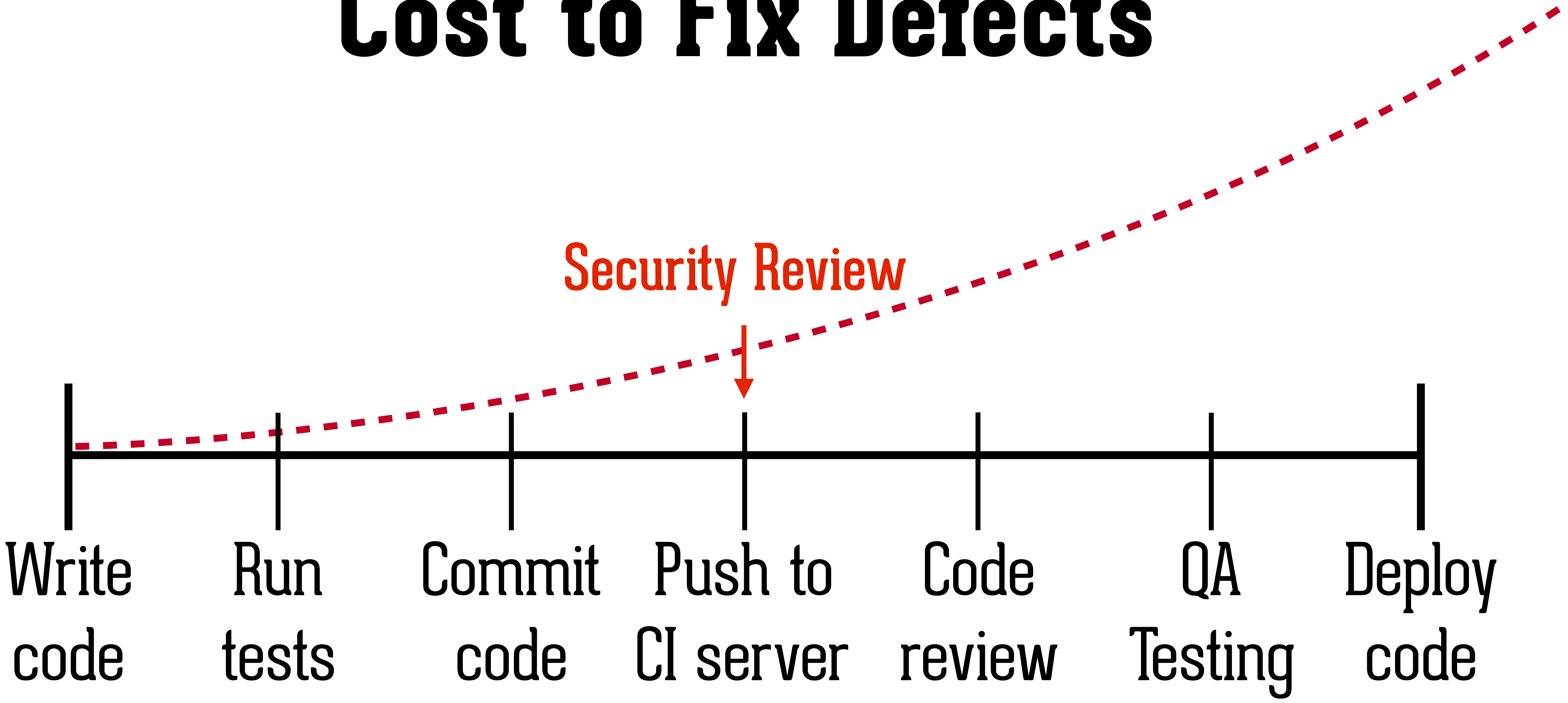
Cost to Fix Defects



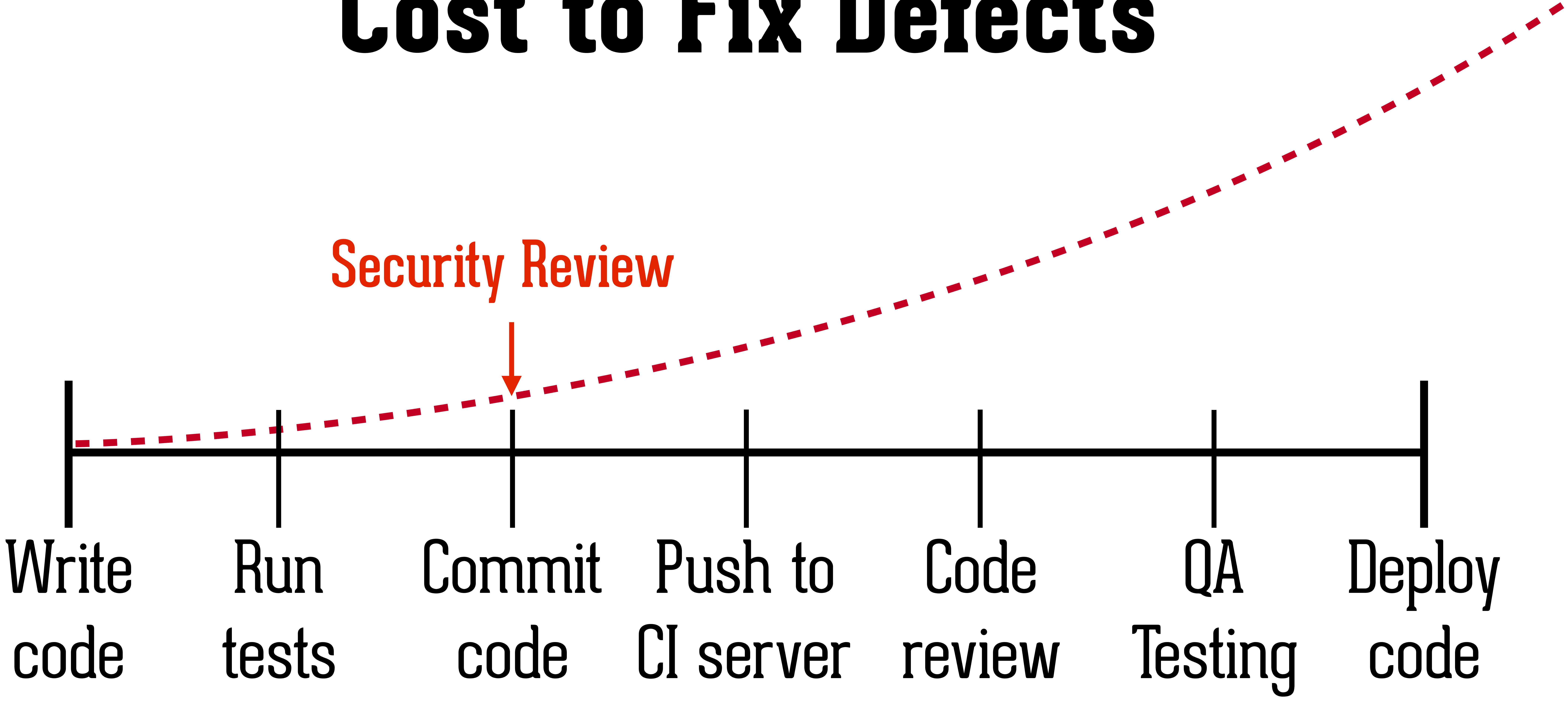
Cost to Fix Defects



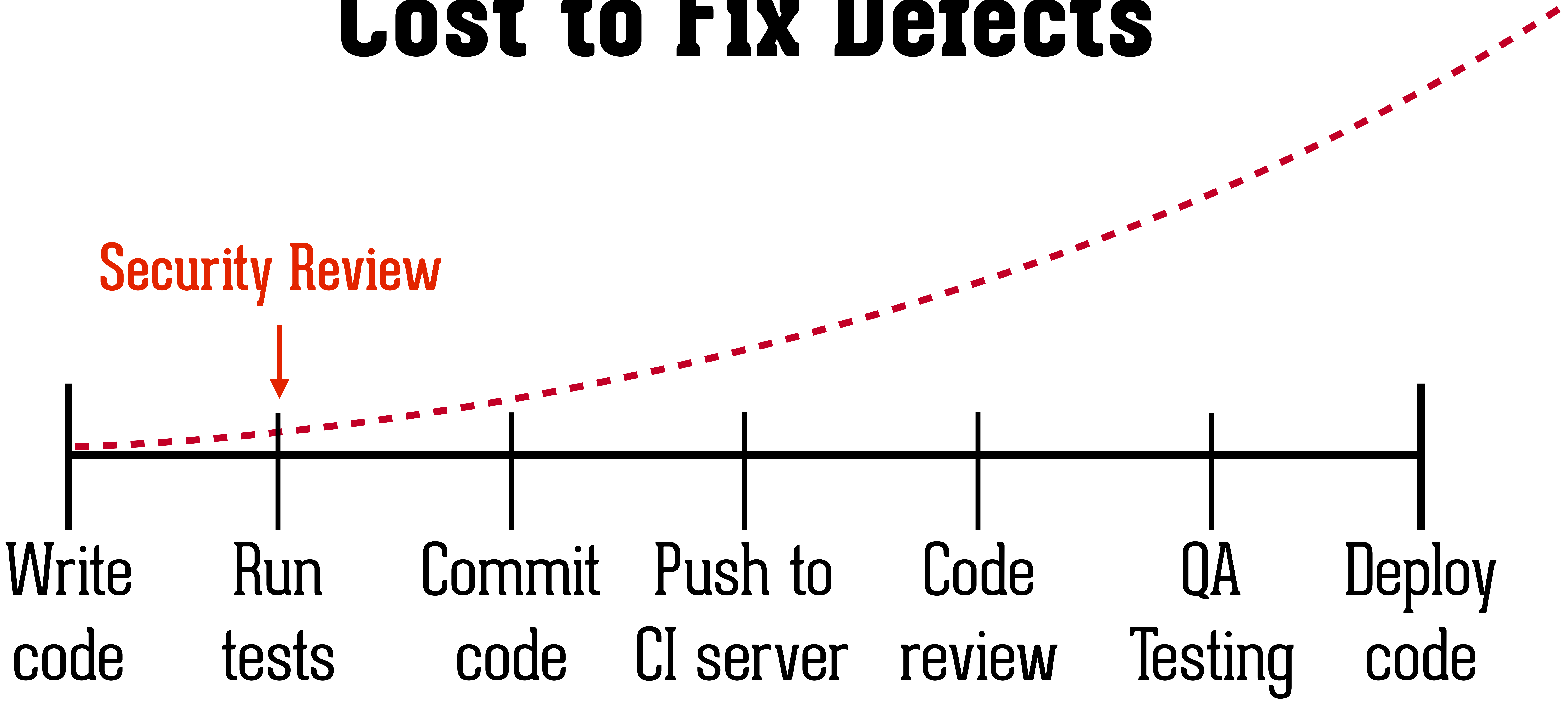
Cost to Fix Defects



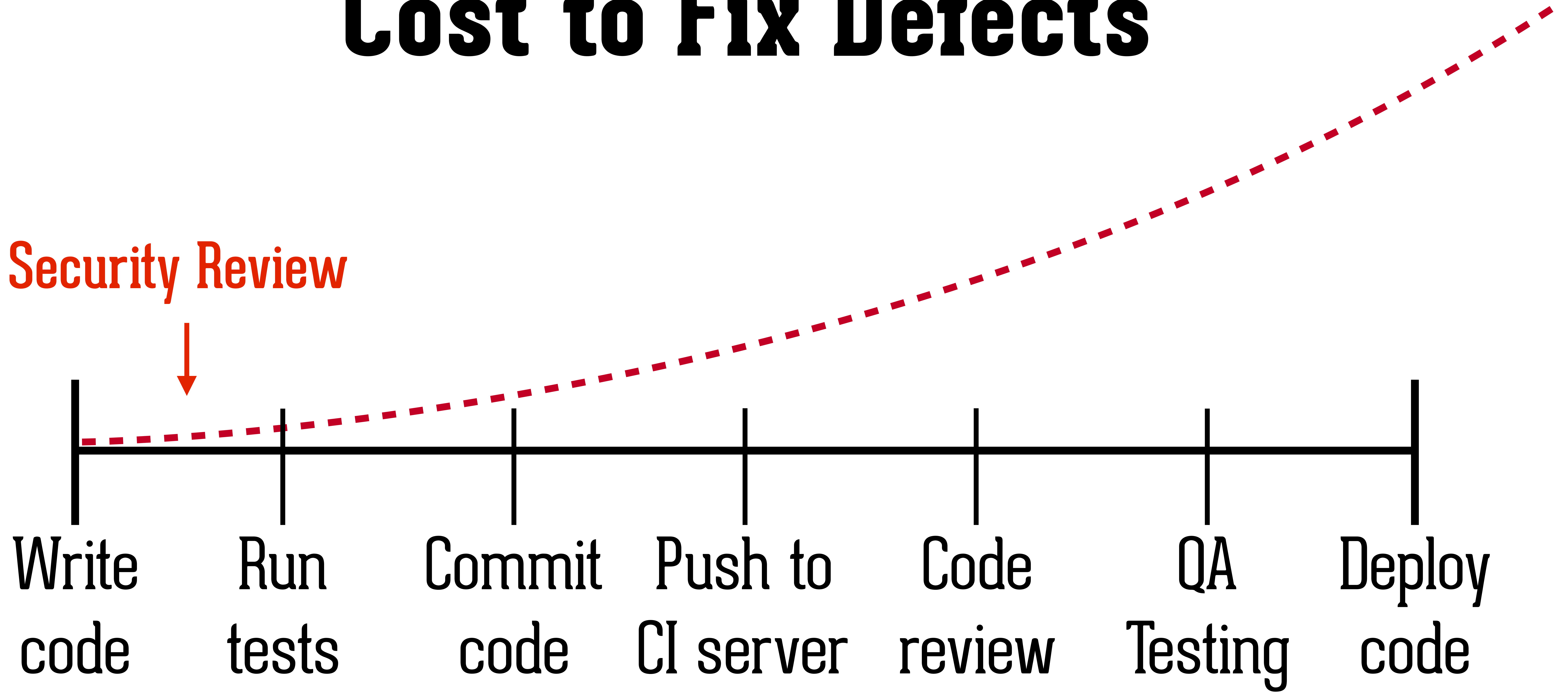
Cost to Fix Defects



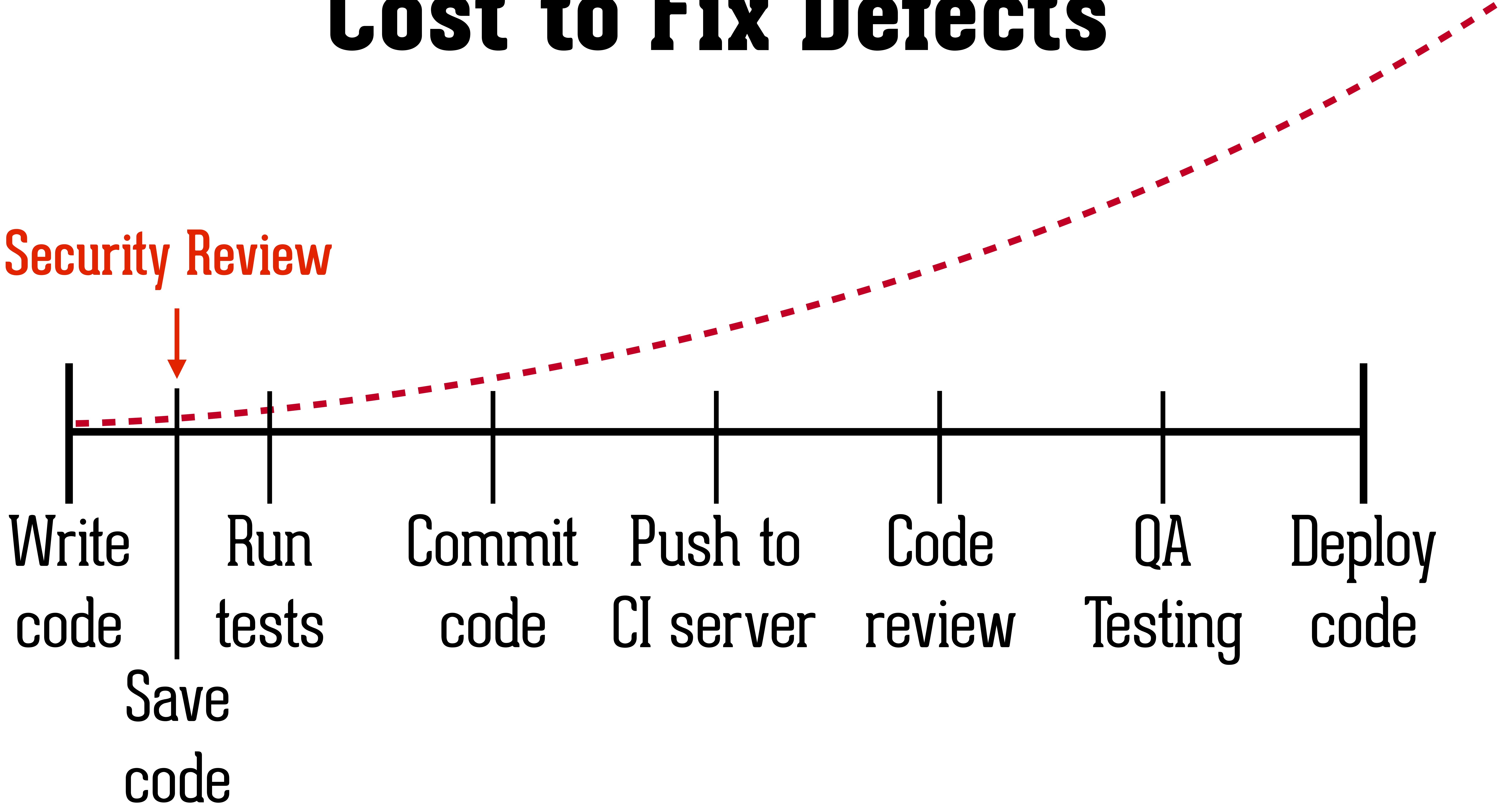
Cost to Fix Defects



Cost to Fix Defects



Cost to Fix Defects



BRAKEMAN

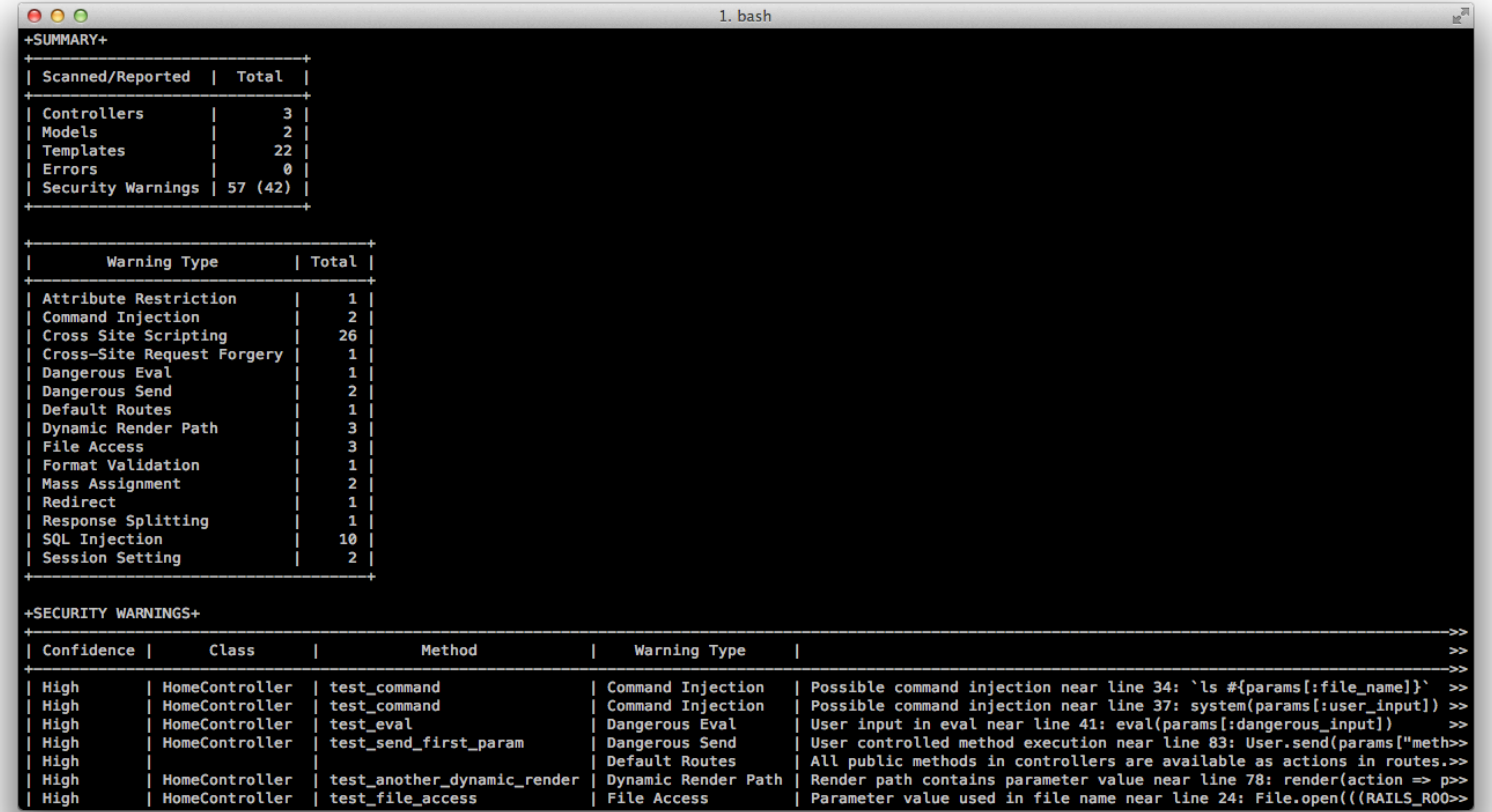
brakemanscanner.org

github.com/presidentbeef/brakeman

@brakeman

“zero configuration” security scanning

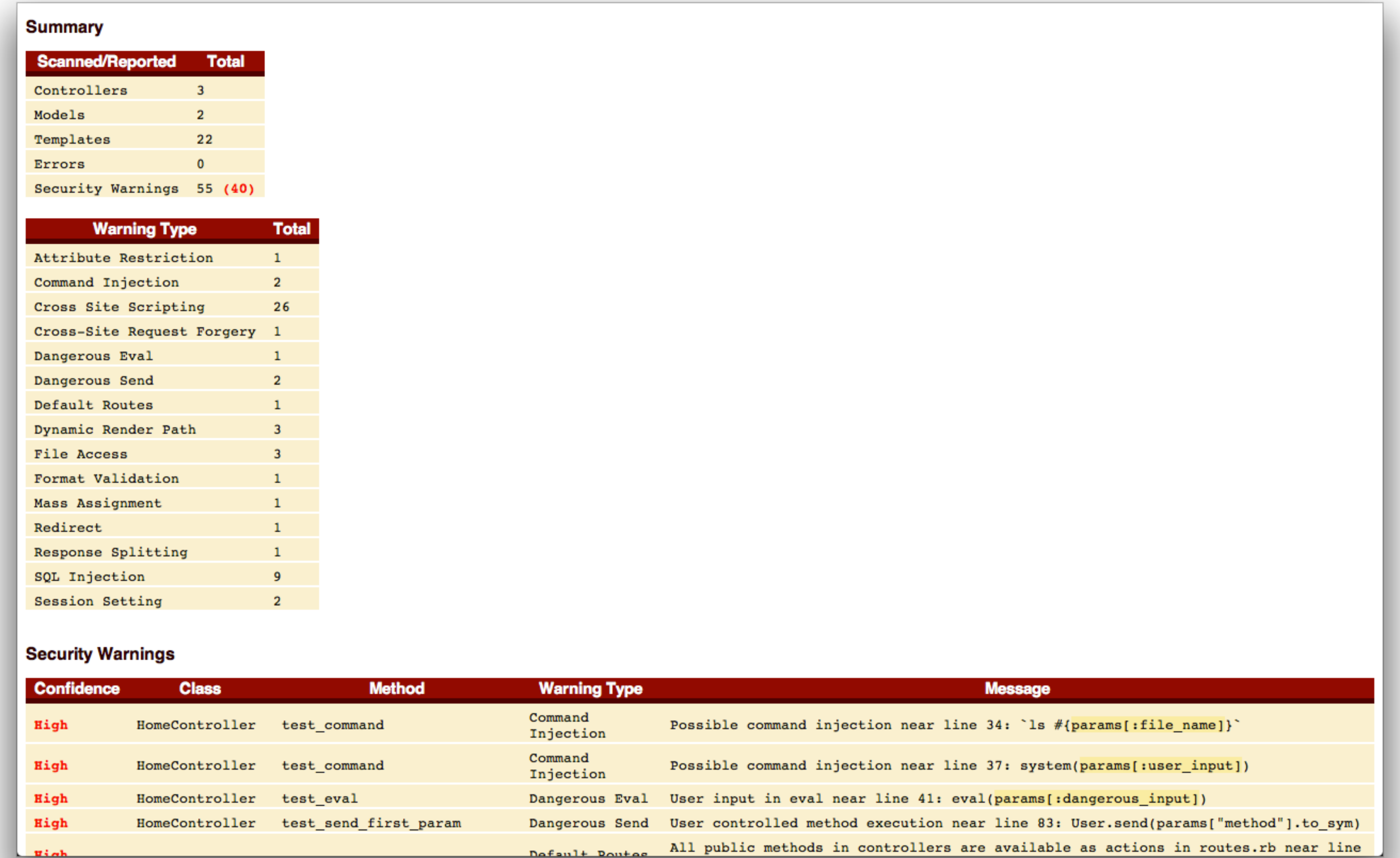
```
gem install brakeman
cd my_rails_app/
brakeman
```



```
+SUMMARY+
+----------------------------+
| Scanned/Reported  |  Total  |
+----------------------------+
| Controllers       |       3 |
| Models            |       2 |
| Templates         |      22 |
| Errors            |       0 |
| Security Warnings | 57 (42) |
+----------------------------+

+-------------------------------------+
|        Warning Type         | Total |
+-------------------------------------+
| Attribute Restriction       |     1 |
| Command Injection           |     2 |
| Cross Site Scripting        |    26 |
| Cross-Site Request Forgery  |     1 |
| Dangerous Eval              |     1 |
| Dangerous Send              |     2 |
| Default Routes              |     1 |
| Dynamic Render Path         |     3 |
| File Access                 |     3 |
| Format Validation           |     1 |
| Mass Assignment             |     2 |
| Redirect                    |     1 |
| Response Splitting          |     1 |
| SQL Injection               |    10 |
| Session Setting             |     2 |
+-------------------------------------+

+SECURITY WARNINGS+
+---------------------------------------------------------------------------------------------------------------------------------------------------------------->>
| Confidence |       Class      |            Method            |     Warning Type      |                                                                      >>
+---------------------------------------------------------------------------------------------------------------------------------------------------------------->>
| High       | HomeController   | test_command                 | Command Injection     | Possible command injection near line 34: `ls #{params[:file_name]}`  >>
| High       | HomeController   | test_command                 | Command Injection     | Possible command injection near line 37: system(params[:user_input]) >>
| High       | HomeController   | test_eval                    | Dangerous Eval        | User input in eval near line 41: eval(params[:dangerous_input])      >>
| High       | HomeController   | test_send_first_param        | Dangerous Send        | User controlled method execution near line 83: User.send(params["meth>>
| High       |                  |                              | Default Routes        | All public methods in controllers are available as actions in routes.>>
| High       | HomeController   | test_another_dynamic_render  | Dynamic Render Path   | Render path contains parameter value near line 78: render(action => p>>
| High       | HomeController   | test_file_access             | File Access           | Parameter value used in file name near line 24: File.open(((RAILS_ROO>>
gem install brakeman
cd my_rails_app/
brakeman -o report.html
```



Summary

Scanned/Reported	Total
Controllers	3
Models	2
Templates	22
Errors	0
Security Warnings	55 (40)

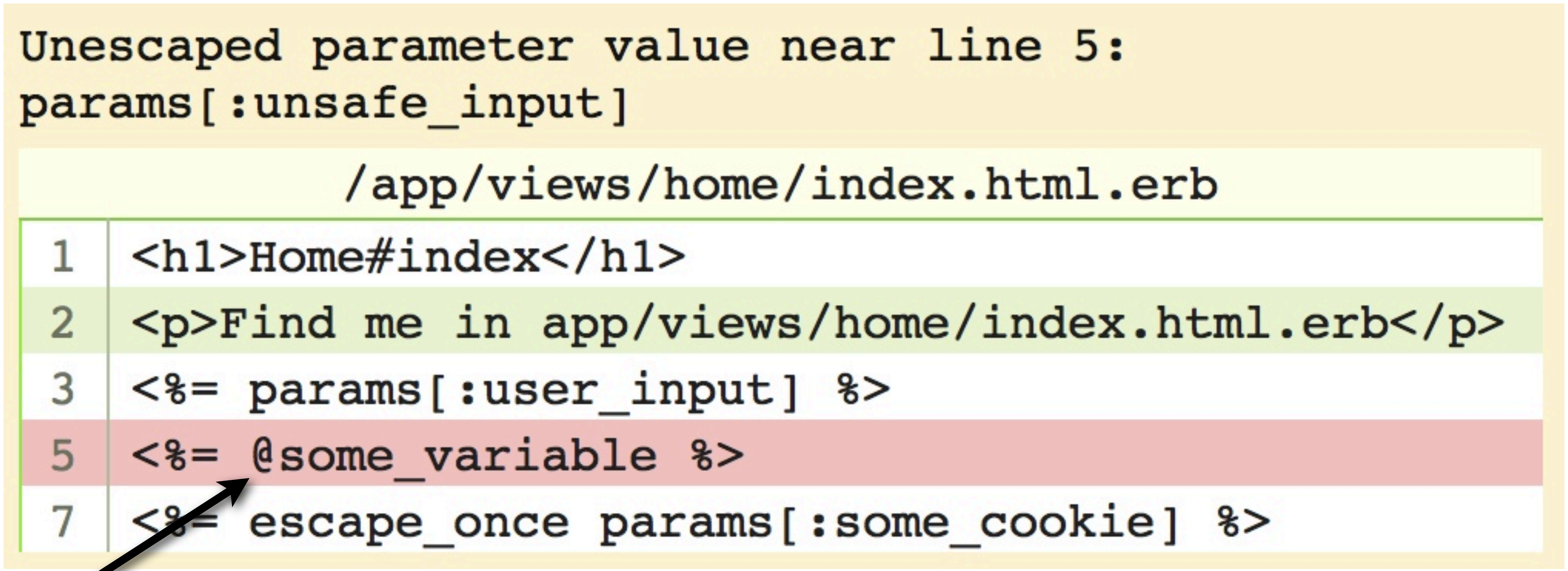
Warning Type	Total
Attribute Restriction	1
Command Injection	2
Cross Site Scripting	26
Cross-Site Request Forgery	1
Dangerous Eval	1
Dangerous Send	2
Default Routes	1
Dynamic Render Path	3
File Access	3
Format Validation	1
Mass Assignment	1
Redirect	1
Response Splitting	1
SQL Injection	9
Session Setting	2

Security Warnings

Confidence	Class	Method	Warning Type	Message
High	HomeController	test_command	Command Injection	Possible command injection near line 34: `ls #{params[:file_name]}`
High	HomeController	test_command	Command Injection	Possible command injection near line 37: system(params[:user_input])
High	HomeController	test_eval	Dangerous Eval	User input in eval near line 41: eval(params[:dangerous_input])
High	HomeController	test_send_first_param	Dangerous Send	User controlled method execution near line 83: User.send(params["method"].to_sym)
High			Default Routes	All public methods in controllers are available as actions in routes.rb near line

"Confidence"
View
Warning Type
Render Location
Line Number
Code Snippet

High	home/index (HomeController#test_render)	Cross Site Scripting	Unescaped parameter value near line 5: params[:unsafe_input]



```
Unescaped parameter value near line 5:
params[:unsafe_input]
```

```
/app/views/home/index.html.erb

1  <h1>Home#index</h1>
2  <p>Find me in app/views/home/index.html.erb</p>
3  <%= params[:user_input] %>
5  <%= @some_variable %>
7  <%= escape_once params[:some_cookie] %>
```

Line 5

Static Analysis Detour

Static Analysis

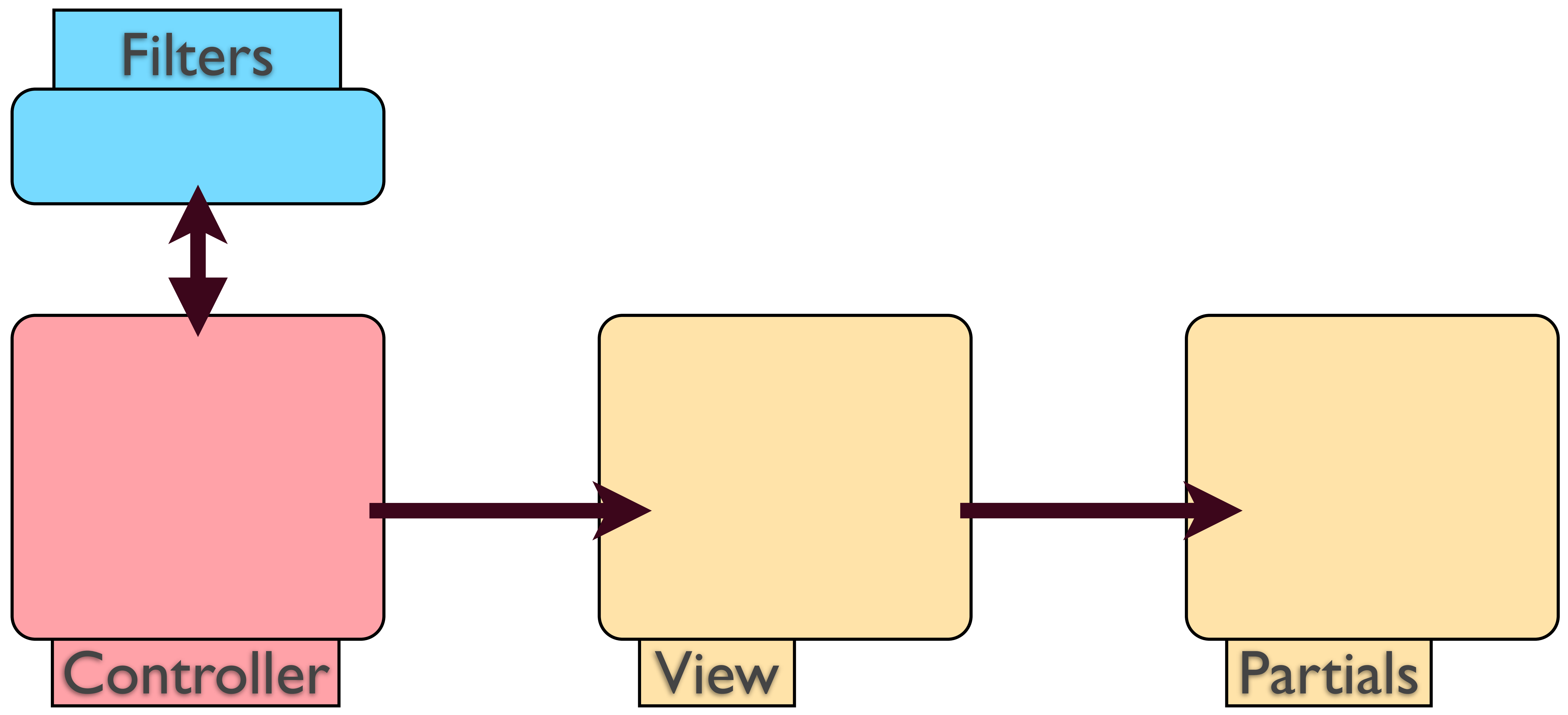
Anything that can be determined about a program without actually executing it

“But Ruby is way too dynamic for that!”

```
eval(File.read(gets.strip))
```

We don’t have to know everything

Most of the Action Happens Here



Start Simple:
User Input in Views

```
<%= params[:user][:name] %>
```

View

Next: From Controllers to Views

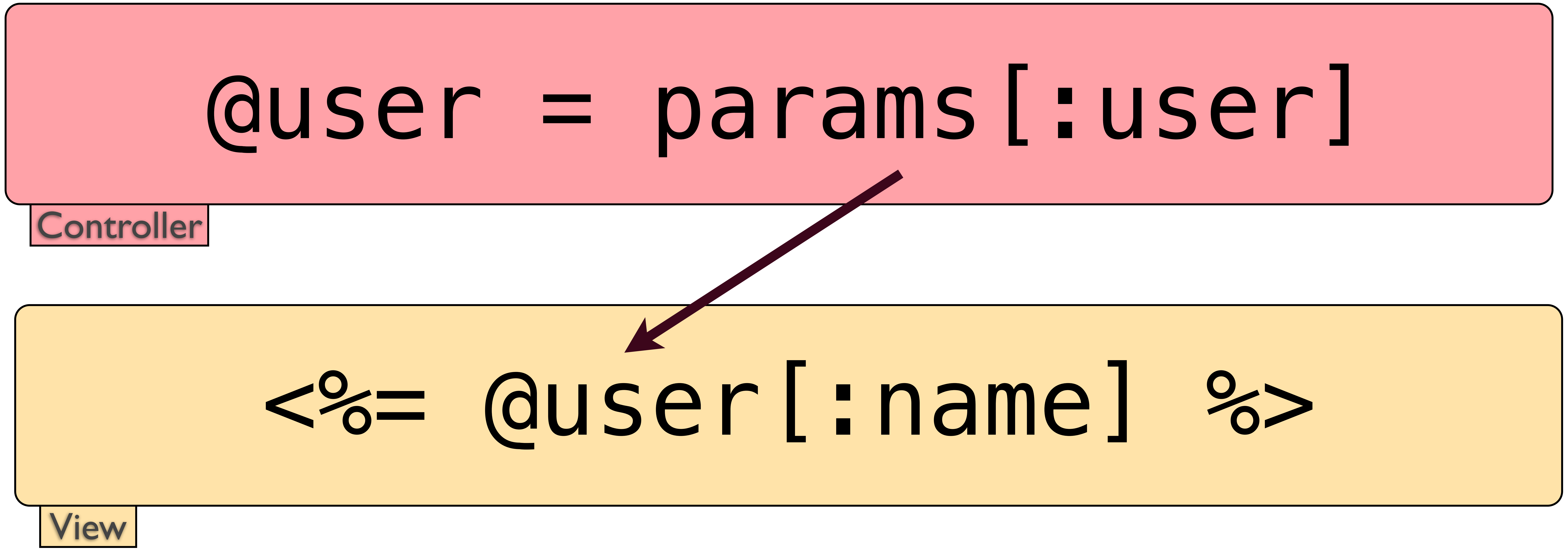
```ruby
@user = params[:user]
```

Controller

```erb
<%= @user[:name] %>
```

View

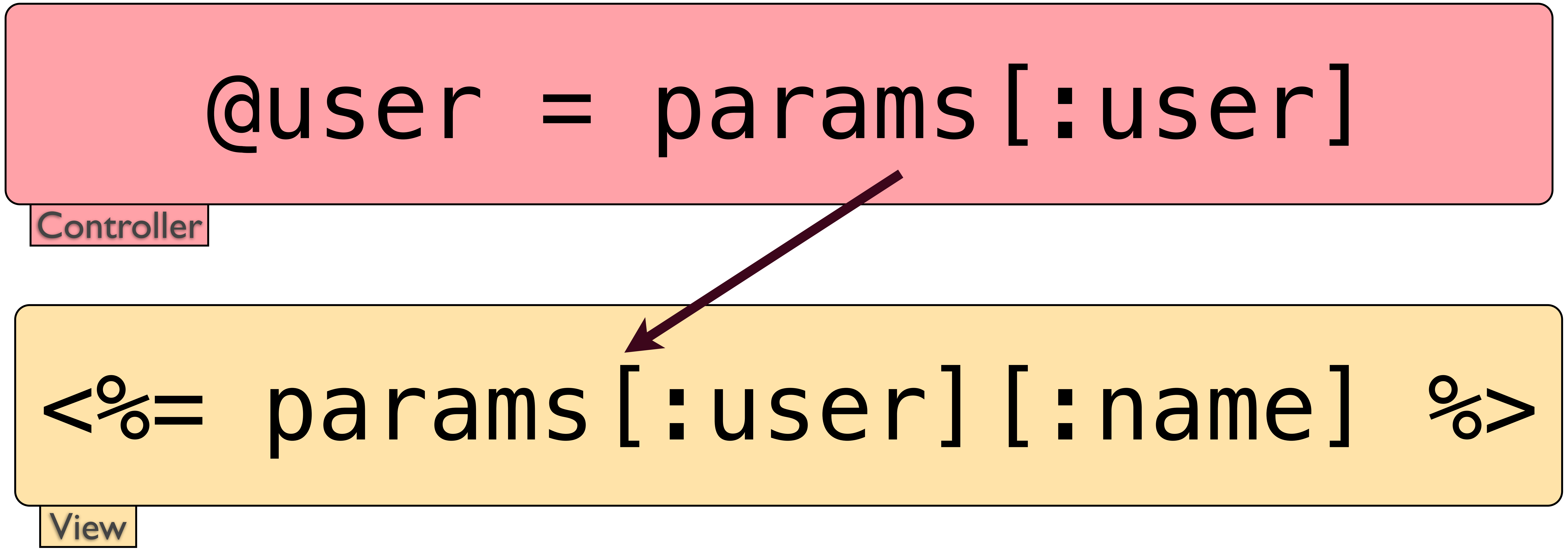
Next: From Controllers to Views



```
@user = params[:user]
```

```
<%= @user[:name] %>
```

Next: From Controllers to Views



```
@user = params[:user]
```

```
<%= params[:user][:name] %>
```

Really Simple Data Flow

```
user = params[:user]
user_id = user[:id]
@current_user = User.find(user_id)
@current_user.update_attributes(user)
```

Really Simple Data Flow

```
user = params[:user]
user_id = user[:id]
@current_user = User.find(user_id)
@current_user.update_attributes(user)
```

Really Simple Data Flow

```
user = params[:user]
user_id = params[:id]
@current_user = User.find(user_id)
@current_user.update_attributes(user)
```

Really Simple Data Flow

```
user = params[:user]
user_id = params[:id]
@current_user = User.find(user_id)
@current_user.update_attributes(user)
```

Really Simple Data Flow

```
user = params[:user]
user_id = params[:id]
@current_user = User.find(params[:user][:id])
@current_user.update_attributes(user)
```

Really Simple Data Flow

```
user = params[:user]
user_id = params[:id]
@current_user = User.find(params[:user][:id])
@current_user.update_attributes(user)
```

Really Simple Data Flow

```
user = params[:user]
user_id = params[:id]
@current_user = User.find(params[:user][:id])
User.find(params[:user][:id]).update_attributes(user)
```

Really Simple Data Flow

```
user = params[:user]
user_id = params[:id]
@current_user = User.find(params[:user][:id])
User.find(params[:user][:id]).update_attributes(user)
```

Really Simple Data Flow

```
user = params[:user]
user_id = params[:id]
@current_user = User.find(params[:user][:id])
User.find(params[:user][:id]).update_attributes(params[:user])
```

Really Simple Data Flow

```
user = params[:user]
user_id = params[:id]
@current_user = User.find(params[:user][:id])
User.find(params[:user][:id]).update_attributes(params[:user])
```

$\uparrow$

Mass

Assignment

Brakeman Can Detect...

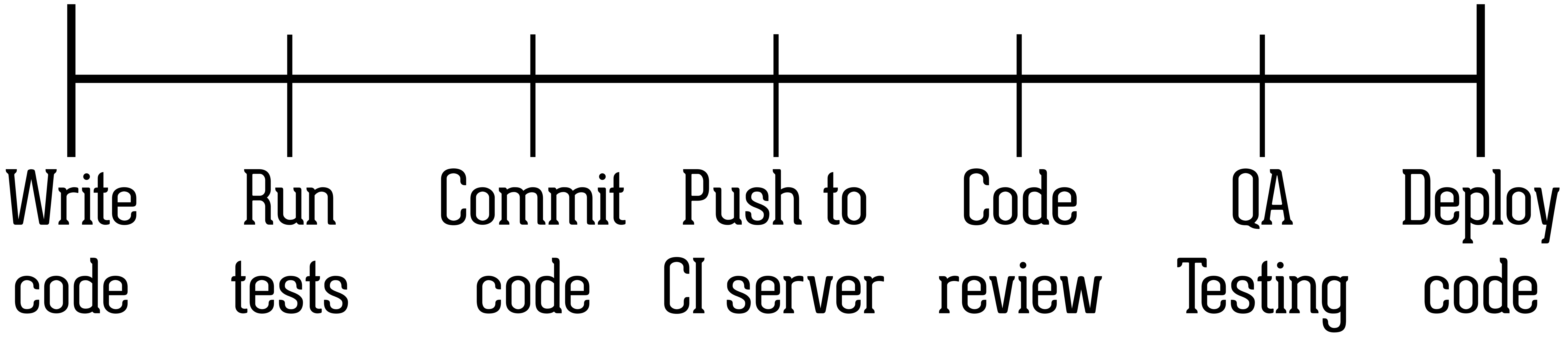
- Cross site scripting
- SQL injection
- Command injection
- Unrestricted mass assignment
- Unprotected redirects
- Unsafe file access
- Insufficient model validation
- Version-specific security issues
- Dangerous use of eval
- Dangerous use of send
- Default routes
- Dynamic render paths
- ...and more!

Performance

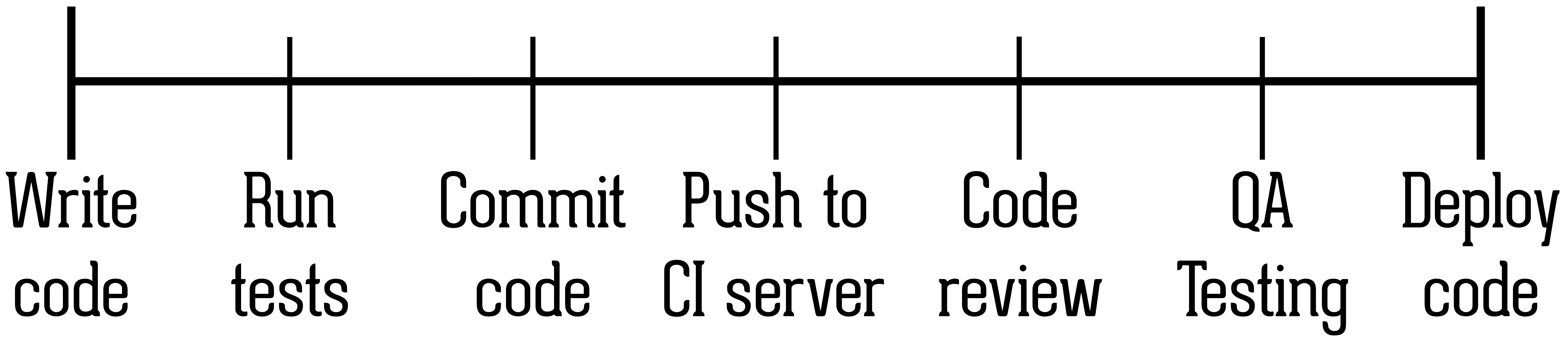
Twitter Main App	< 2m
nventory [66c, 58m, 688t]	~1m
Redmine [50c, 77m, 256t]	~20s
Typo [34c, 47m, 113t]	~5s

Brakeman 1.6.0, Ruby 1.9.3-p125

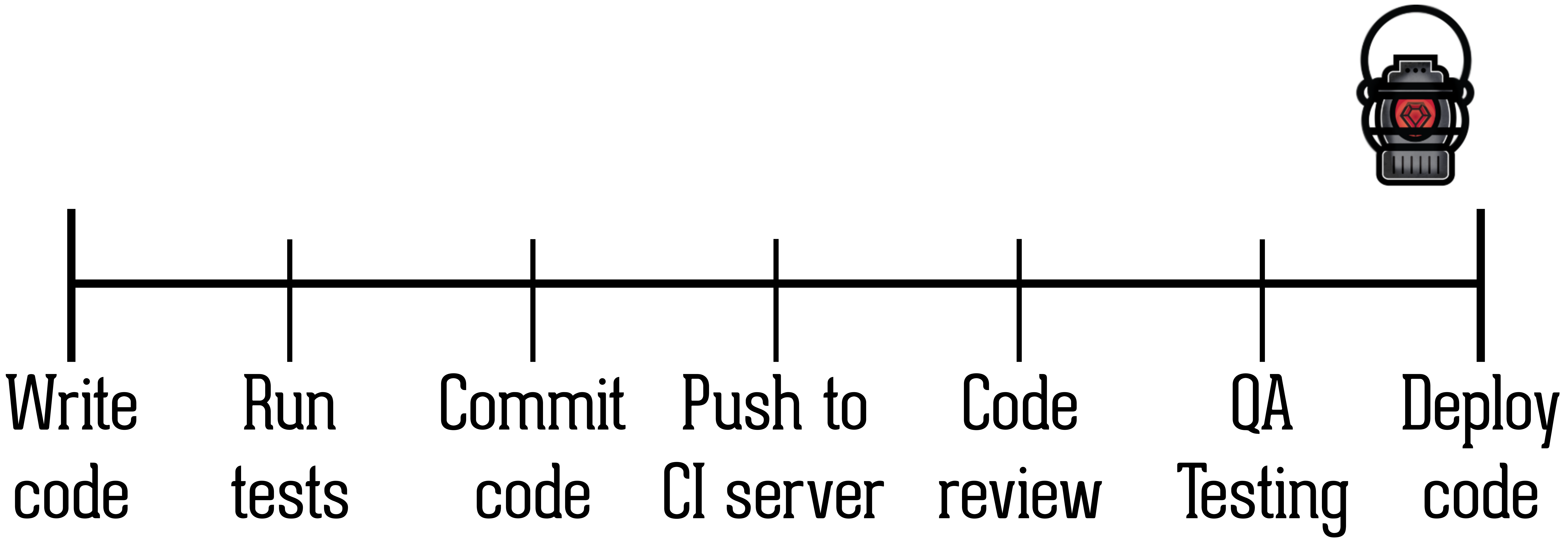
Back to SDLC


Write code
Run tests
Commit code
Push to CI server
Code review
QA Testing
Deploy code

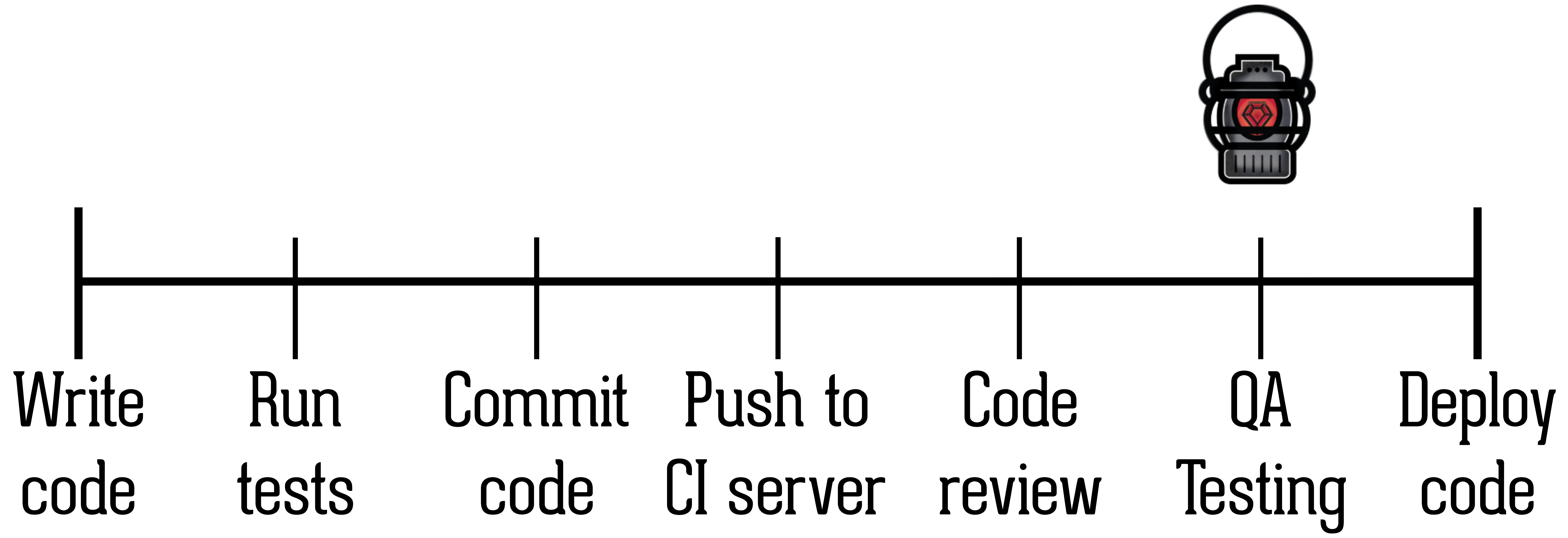
Run Brakeman Anytime



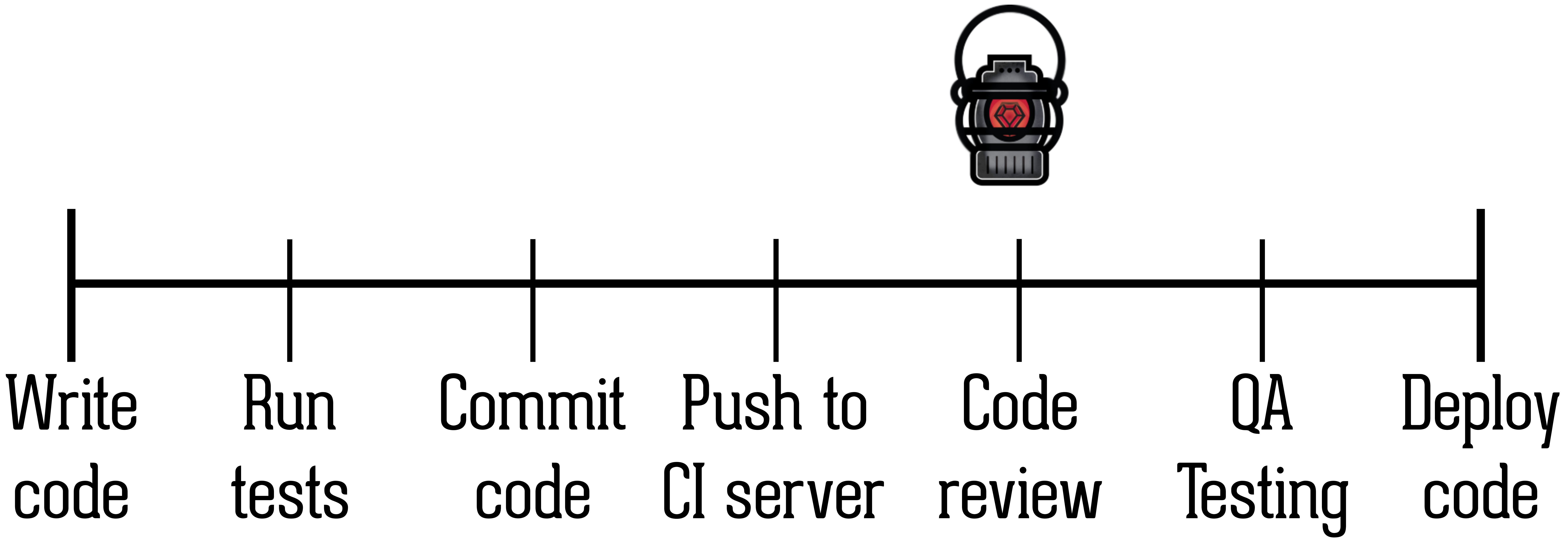
Run Brakeman Anytime



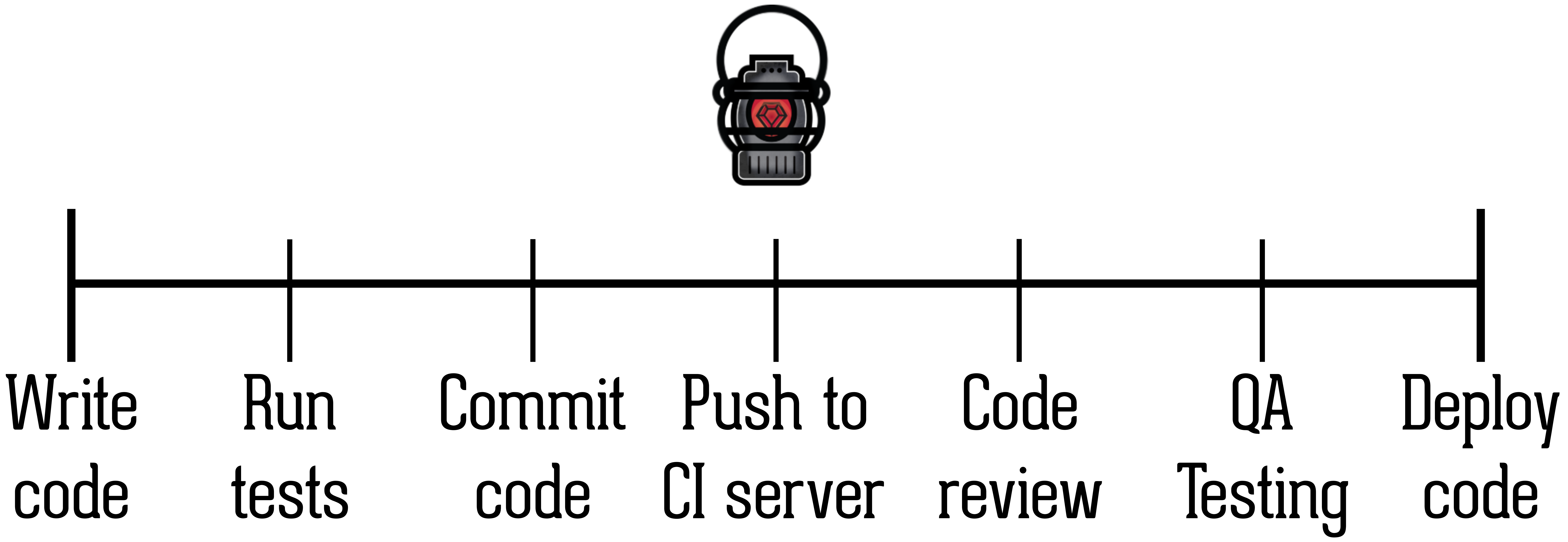
Run Brakeman Anytime



Run Brakeman Anytime



Run Brakeman Anytime

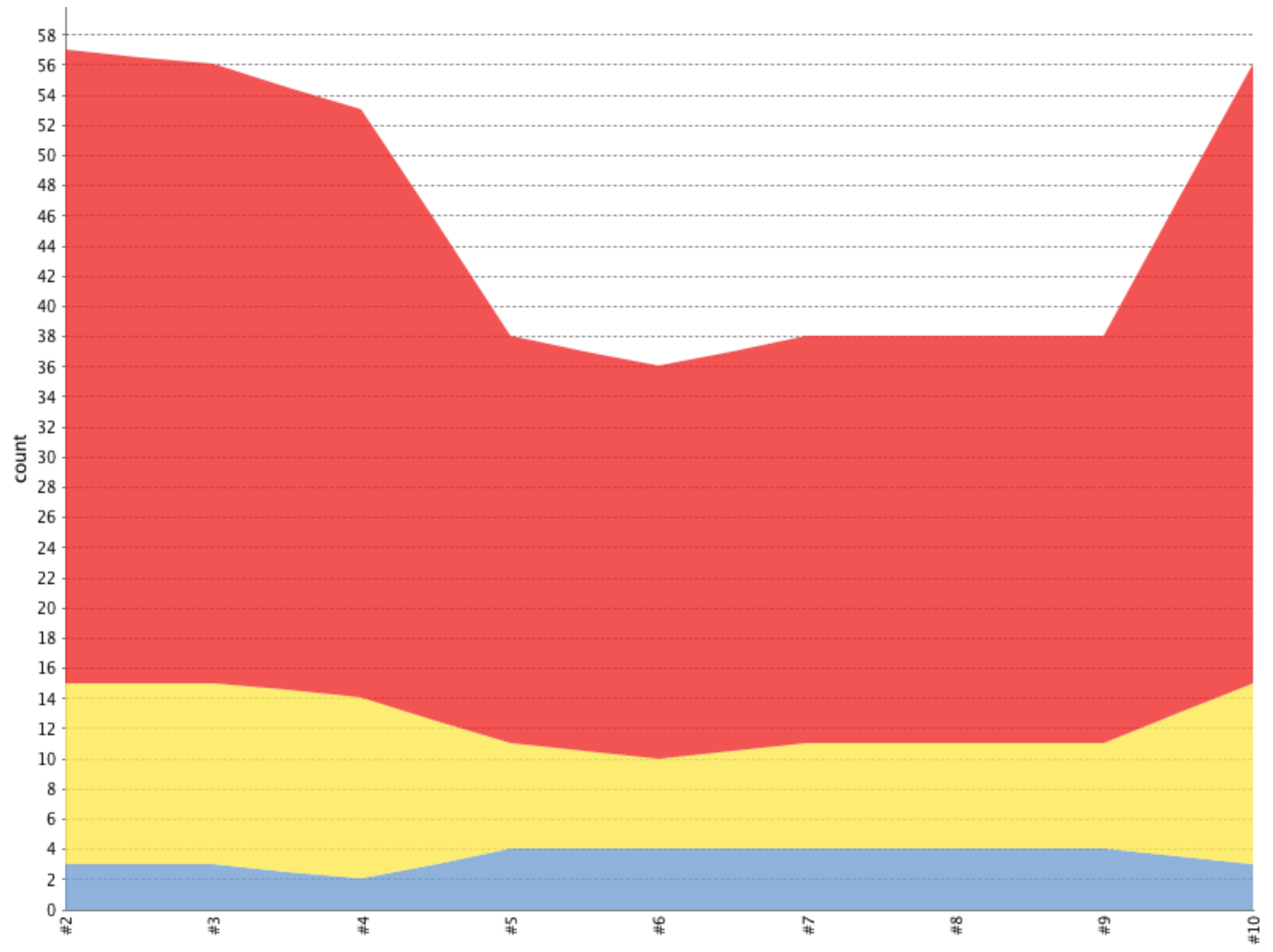


Brakeman + Jenkins

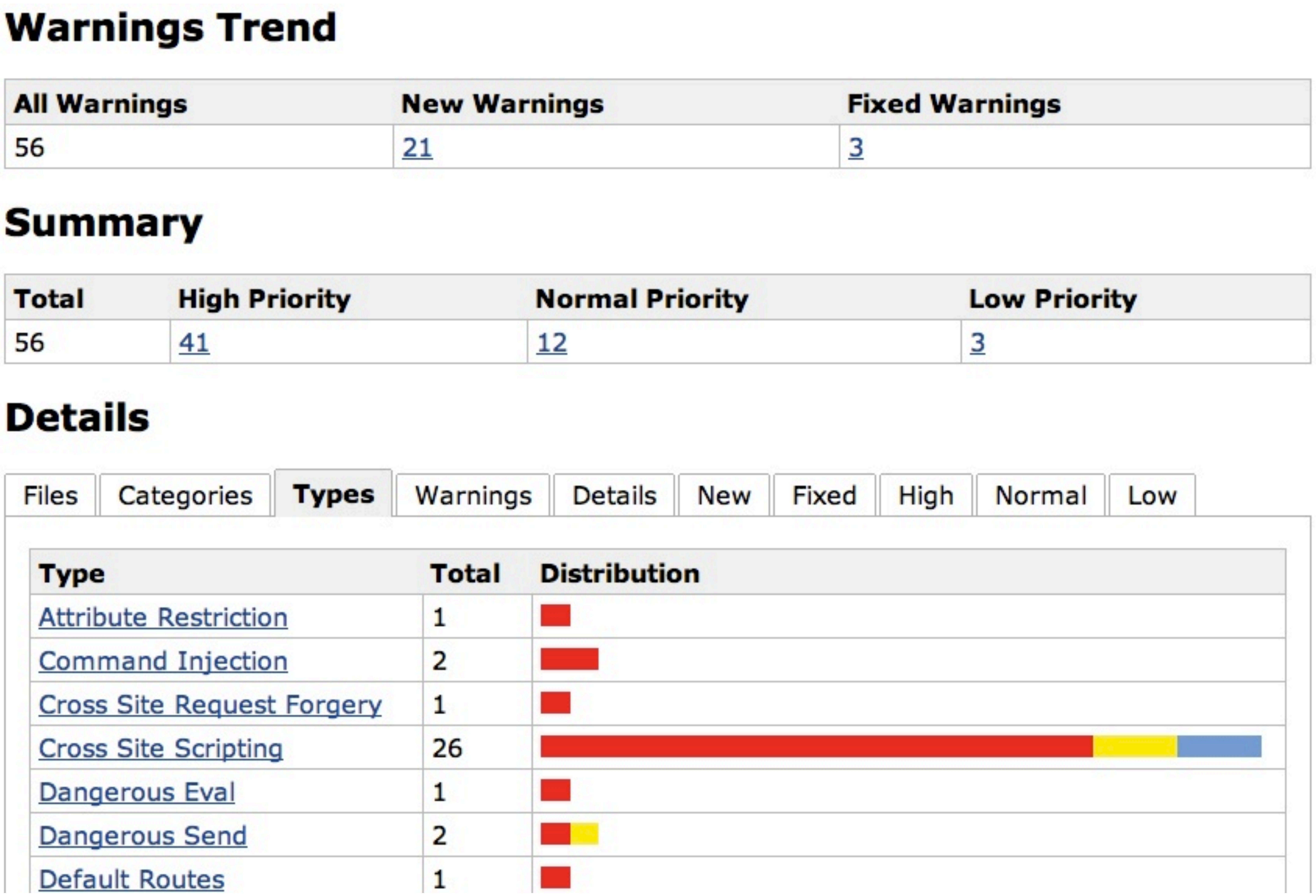
jenkins-ci.org

open source CI server

Brakeman + Jenkins



Brakeman + Jenkins



Warnings Trend

All Warnings	New Warnings	Fixed Warnings
56	21	3

Summary

Total	High Priority	Normal Priority	Low Priority
56	41	12	3

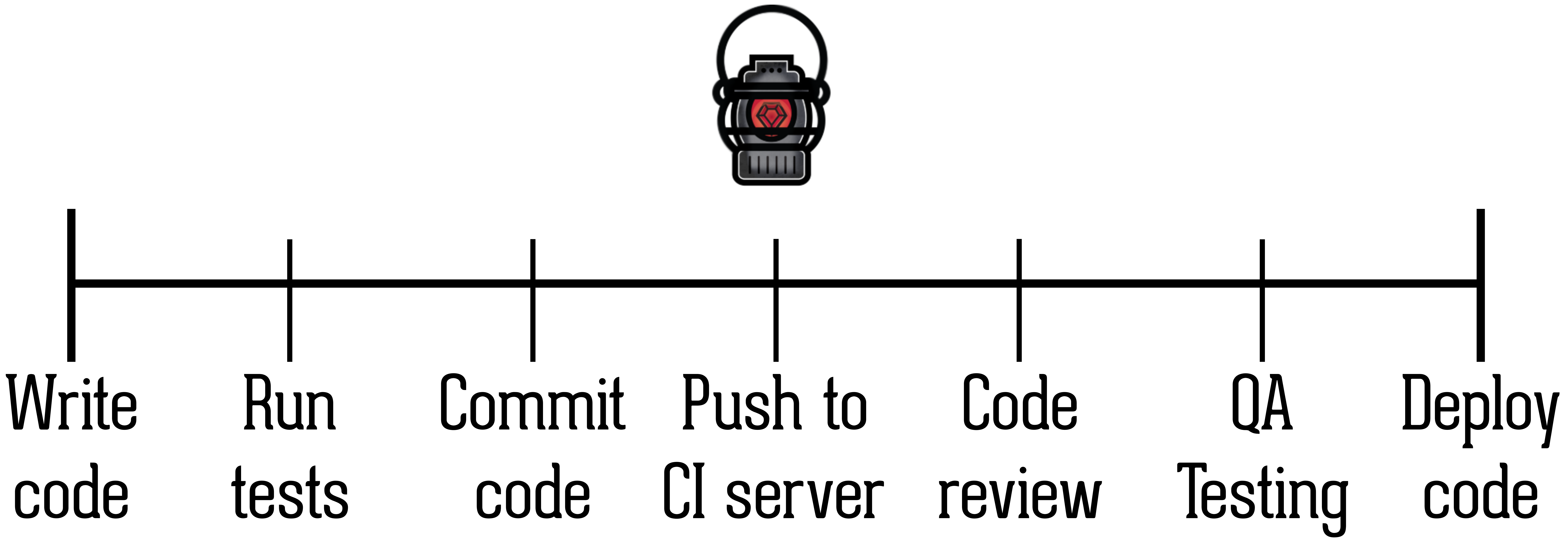
Details

Type	Total	Distribution
Attribute Restriction	1	🟥
Command Injection	2	🟥
Cross Site Request Forgery	1	🟥
Cross Site Scripting	26	🟥🟨🟦
Dangerous Eval	1	🟥
Dangerous Send	2	🟥🟨
Default Routes	1	🟥

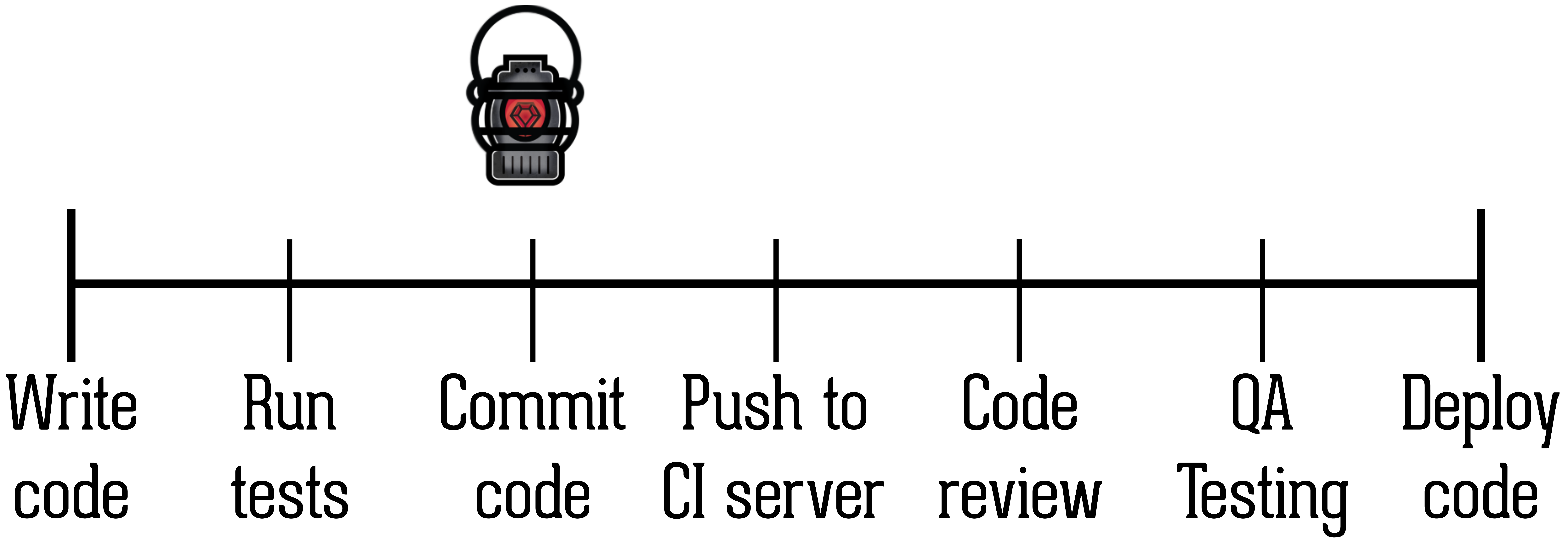
Brakeman Programatically

```
require “brakeman”
Brakeman.run “myapp”
```

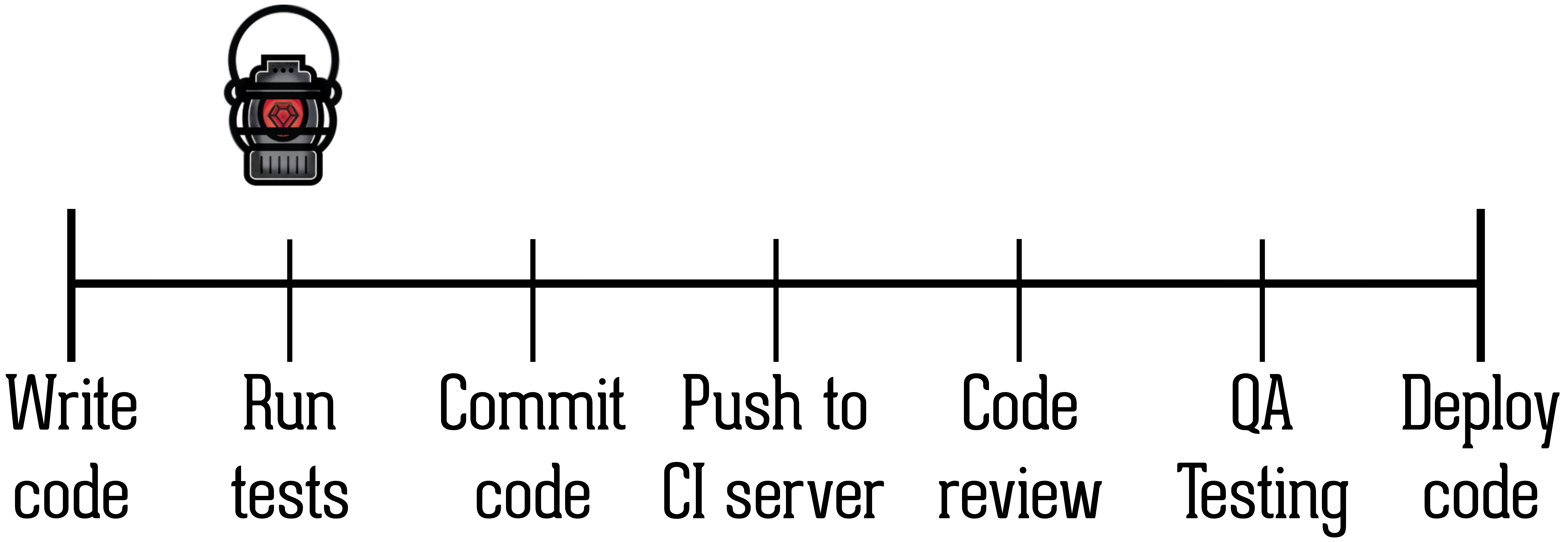
Run Brakeman Anytime



Run Brakeman Anytime



Run Brakeman Anytime



Brakeman + Rake

```
brakeman --rake
rake brakeman:run
```

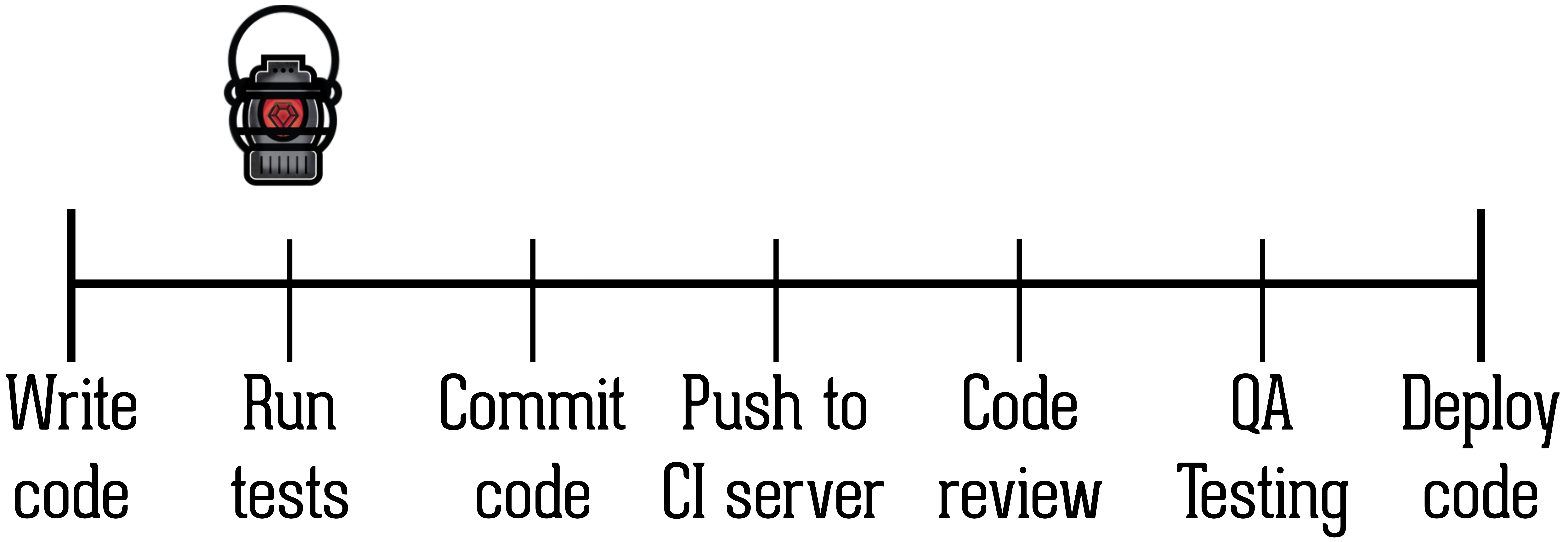
Hardcore Mode

```
brakeman -z
```

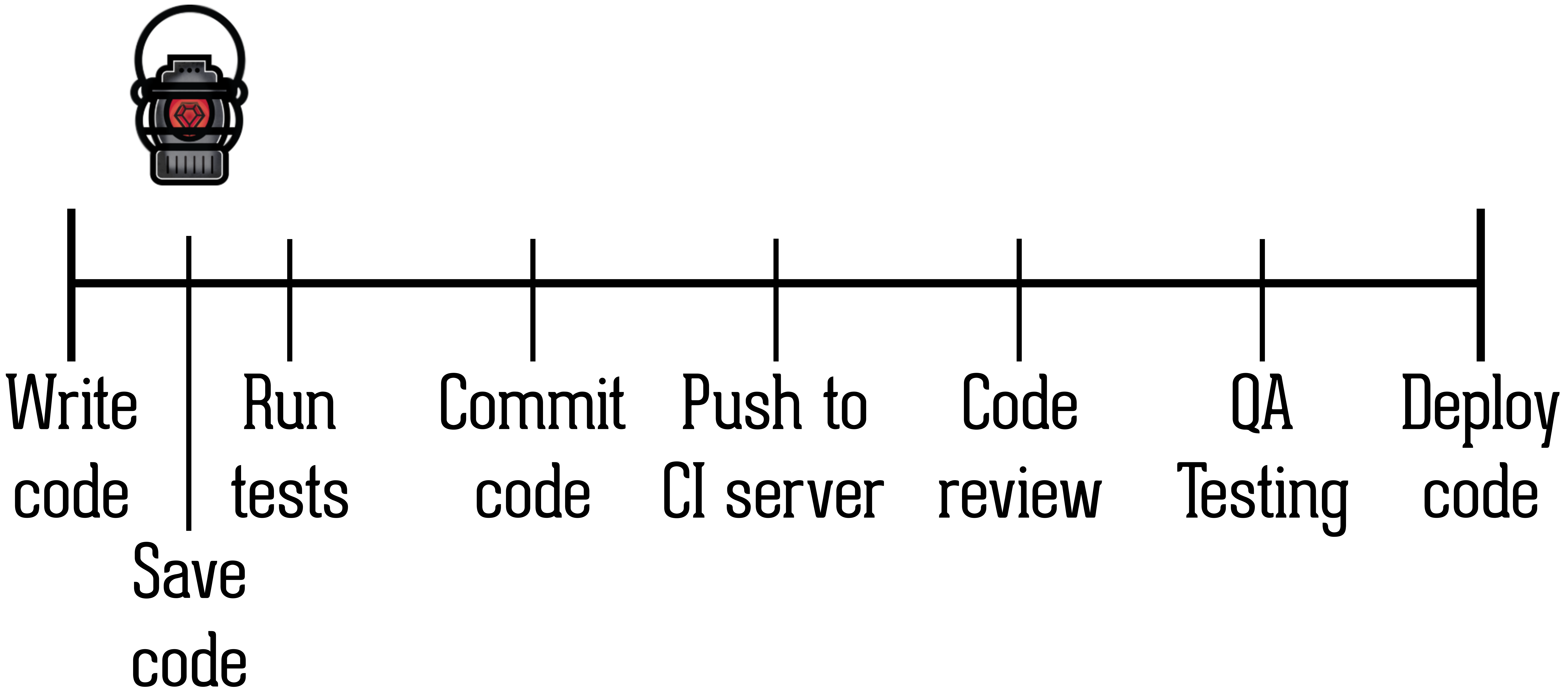
Comparing Brakeman Results

```
brakeman -o report.json
brakeman --compare report.json
```

Brakeman...All the Time?



Brakeman...All the Time?



Fast Rescanning

Brakeman supports fast rescanning of changed files*

Fast Rescanning

*If scan is kept in memory

Brakeman + Guard

```
group :development do
  gem 'guard-brakeman'
end
```

Brakeman + Guard

```
guard init brakeman
guard
```

Brakeman + Guard Demo

http://www.youtube.com/watch?v=CMgYcr9_0Ns

Caveats

warnings != vulnerabilities

zero warnings does not mean zero vulnerabilities

Brakeman is not omniscient

Supports

- Rails 2.x & 3.x
- Ruby 1.8.7 & 1.9.x
- JRuby

brakemanscanner.org

github.com/presidentbeef/brakeman

@brakeman