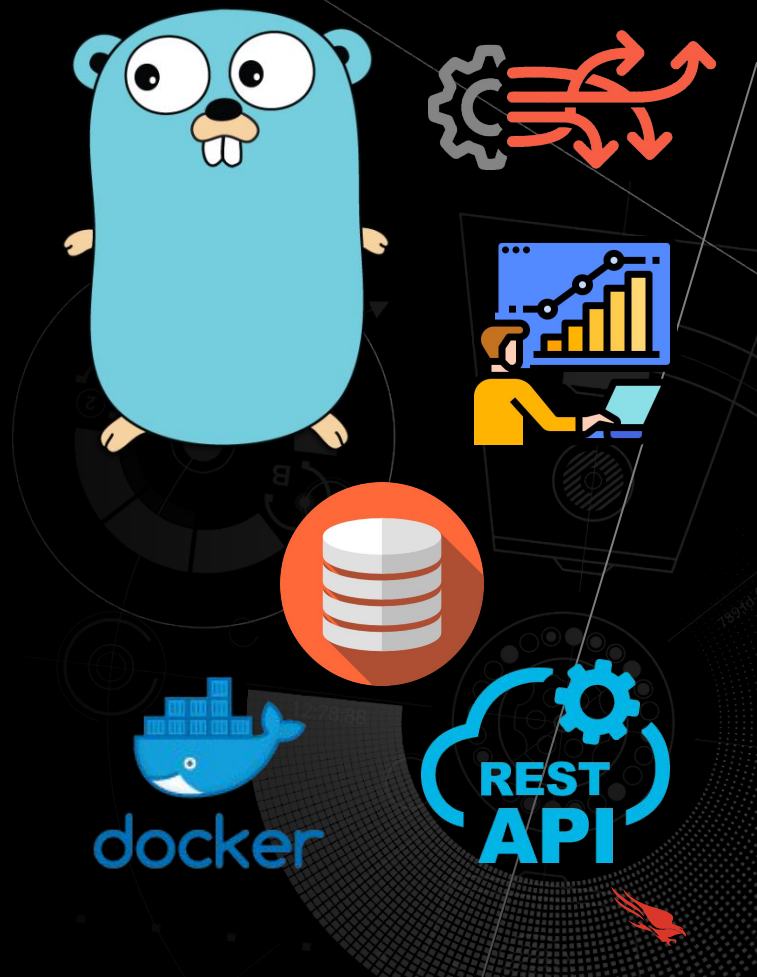
Unit testing, logging and monitoring

CrowdStrike HEROES - Cloud Workshop

CROWDSTRIKE

Date	Topic
July 25 - 16:00	Intro to golang
July 26 - 16:00	Intro to golang (continuation)
July 27 - 16:00	Multithreading
July 28 - 16:00	Rest API
July 29 - 16:00	**Unit testing, logging and monitoring**
August 1 - 16:00	Workshop and Q&A
August 2 - 16:00	Deployments/Docker
August 3 - 16:00	Databases
August 4 - 16:00	Databases extended
August 5 - 13:00	Microservices contest (4h with Awards)



CROWDSTRIKE

Testing

Why test my code?

How do I test my code?

When should I test my code?

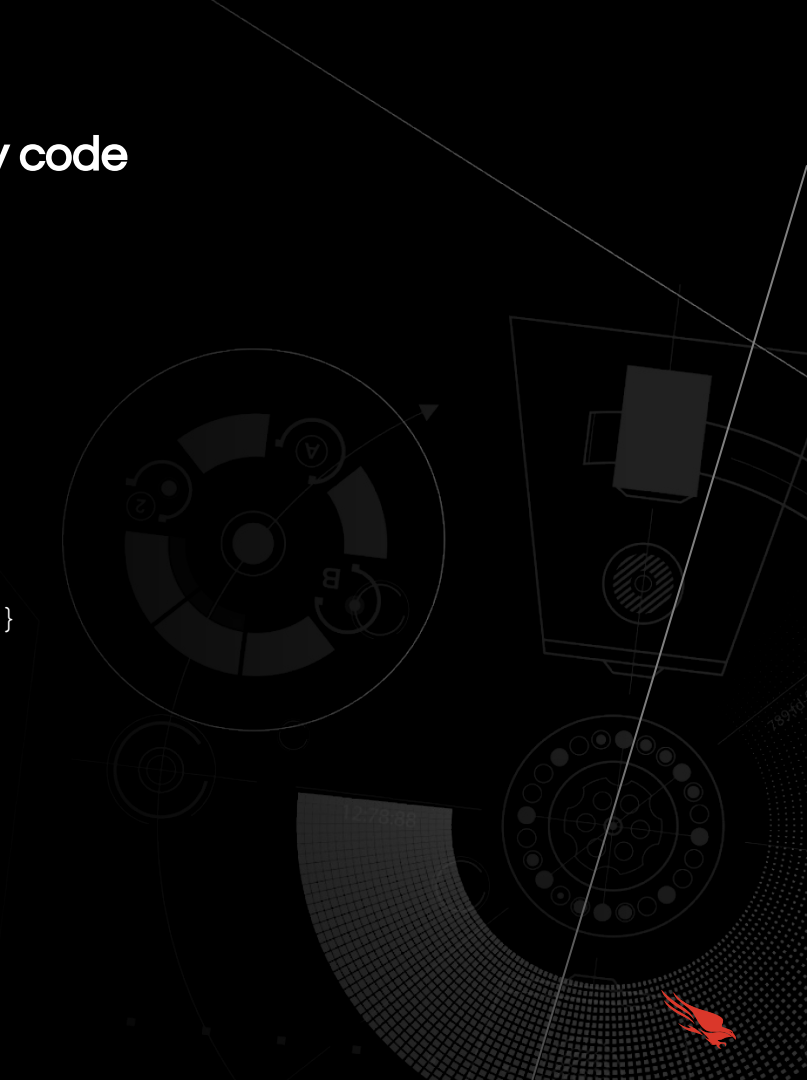
CROWDSTRIKE

Why should I test my code

Bugs can appear no matter how skilled you are

Example:

```
var users map[int]domain.User
users[userID] = domain.User{Name: "TestUser"}
```



CROWDSTRIKE

Testing in practice

Types of testing

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

CROWDSTRIKE

Unit Testing

“Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. This testing methodology is done during the development process by the software developers and sometimes QA staff.”

Pros and cons of unit testing

Pros

- The earlier a problem is identified, the fewer compound errors occur.
- Costs of fixing a problem early can quickly outweigh the cost of fixing it later.
- Debugging processes are made easier.
- Developers can quickly make changes to the code base.

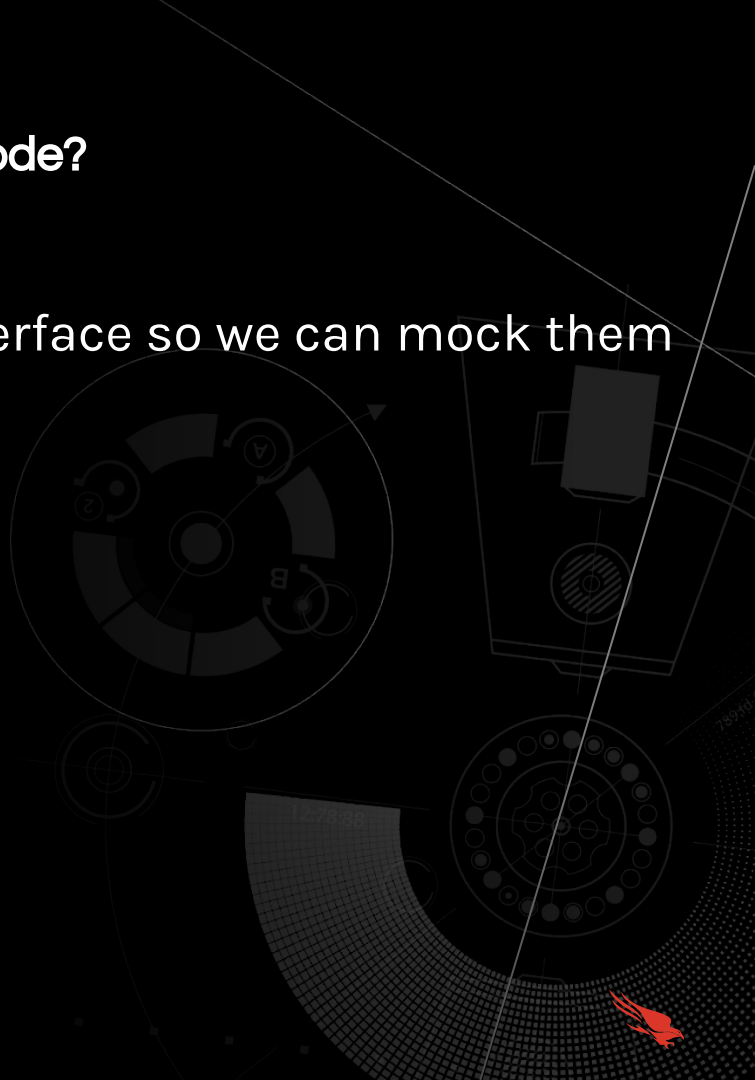
Cons

- Tests will not uncover every bug.
- Unit tests only test sets of data and its functionality—it will not catch errors in integration.
- More lines of test code may need to be written to test one line of code—creating a potential time investment.
- Unit testing may have a steep learning curve, for example, having to learn how to use specific automated software tools.

CROWDSTRIKE

How should I test my code?

- In go all components should have an interface so we can mock them
- We should try to get maximum coverage



CROWDSTRIKE

When to test my code

- Always
- Test driven development
- Smoke tests while service is deployed

CROWDSTRIKE

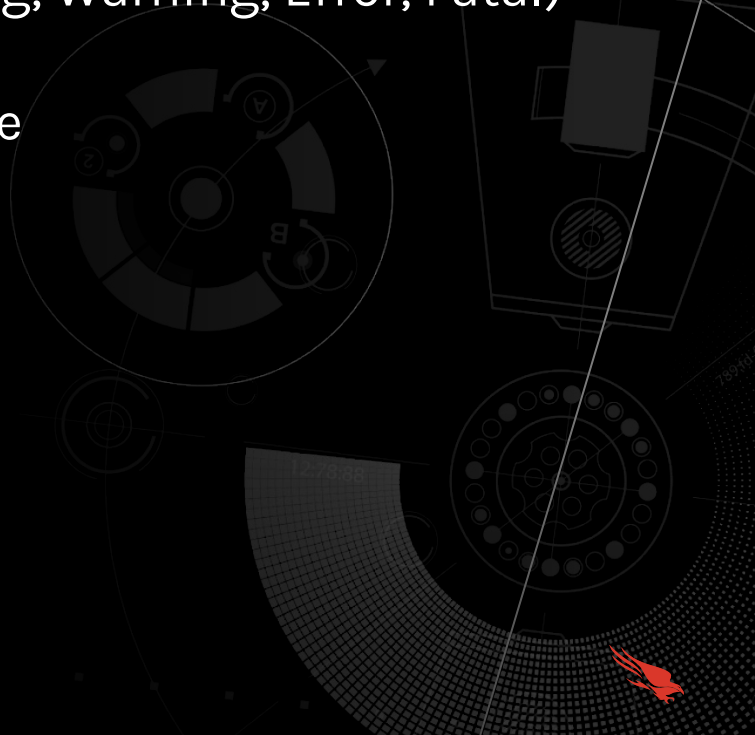
Logging

- Logs help us understand what happens in our service
- We should be able to track a behaviour
- We should be able to see errors

CROWDSTRIKE

Monitoring

- Can be done using logs (Trace, Info, Debug, Warning, Error, Fatal)
- Crucial for understanding behaviours
- Can tell us something about performance



CROWDSTRIKE

Monitoring

- Can be done using metrics
- Is used to set alerts when something goes wrong
- Usually is done using some UI tool such as Grafana
- More will be presented at the docker workshop

CROWDSTRIKE

Mocking in go
Demo

```
// MockStorage is a mock of Storage interface.
type MockStorage struct {
	ctrl     *gomock.Controller
	recorder *MockStorageMockRecorder
}

// GetContent mocks base method.
func (m *MockStorage) GetContent(id string) (string, error) {
	// ctrl field stores this call
}
// GetContent indicates an expected call of GetContent.
func (mr *MockStorageMockRecorder) GetContent(id interface{}) *gomock.Call {
	// this method should be called inside a *_test.go file
	// gomock.Call can represent the number of calls and the expected return value
}
```

```
type Storage interface {
	GetContent(id string) (string, error)
	WriteContent(id string, content string) error
}
```

CROWDSTRIKE

Any questions?