- So right now we just hav this very basic `ExecuteTask`, just prints, really does not do anything, let's change that

First thing is we want to do instantiate empty "generic" AgentTaskResult struct, custom type we just created in previous lesson.

```
var result models.AgentTaskResult
```

Now we add this, which will locate the correct orchestrator based on keyword

```
orchestrator, found := agent.commandOrchestrators[job.Command]
```

Obvs, since we only have 1 command atm won't make much difference, but when there are multiple this allows the correct orchestrator to be called based on `job.Command`

Now code that will actually call the orchestrator, IF it was found above

```
	if found {
		result = orchestrator(agent, job)
	} else {
		log.Printf("|WARN AGENT TASK| Received unknown command: '%s' (ID: %s)", job.Command, job.JobID)
		result = models.AgentTaskResult{
			JobID:   job.JobID,
			Success: false,
			Error:   errors.New("command not found"),
		}
	}
```

So now either way we have a result struct

- If it was found it either succeeded and has the actual result, or there was an error somewhere in execution it has that error
- If it did not find the command here at all we can see here we have error

But now we need to marshall the struct to send it back to server

```
    // Now marshall the result before sending it back
    resultBytes, err := json.Marshal(result)
    if err != nil {
       log.Printf("|❗ ERR AGENT TASK| Failed to marshal result for Task
ID %s: %v", job.JobID, err)
       return // Cannot send result if marshalling fails
    }
```

- And finally we need to send it back
- Now this is different than our "normal" Send() function
- That is for check-ins, and receiving instructions
- This one is specifically for sending back results and receiving confirmation it was received
- This one is called SendResult()

```
    // Now pass it to SendResult()
    log.Printf("|AGENT TASK|-> Sending result for Task ID %s (%d
bytes)...", job.JobID, len(resultBytes))
    err = agent.SendResult(resultBytes)
    if err != nil {
       log.Printf("|❗ ERR AGENT TASK| Failed to send result for Task ID
%s: %v", job.JobID, err)
    }
```

For reference here is the entire function

```
func (agent *Agent) ExecuteTask(job *models.ServerResponse) {
    log.Printf("AGENT IS NOW PROCESSING COMMAND %s with ID %s",
job.Command, job.JobID)

    var result models.AgentTaskResult
```

```go
    orchestrator, found := agent.commandOrchestrators[job.Command]

    if found {
        result = orchestrator(agent, job)
    } else {
        log.Printf("|WARN AGENT TASK| Received unknown command: '%s' (ID: %s)", job.Command, job.JobID)
        result = models.AgentTaskResult{
            JobID:   job.JobID,
            Success: false,
            Error:   errors.New("command not found"),
        }
    }
    // Now marshall the result before sending it back
    resultBytes, err := json.Marshal(result)
    if err != nil {
        log.Printf("|❗ ERR AGENT TASK| Failed to marshal result for Task ID %s: %v", job.JobID, err)
        return // Cannot send result if marshalling fails
    }

    // Now pass it to SendResult()
    log.Printf("|AGENT TASK|-> Sending result for Task ID %s (%d bytes)...", job.JobID, len(resultBytes))
    err = agent.SendResult(resultBytes)
    if err != nil {
        log.Printf("|❗ ERR AGENT TASK| Failed to send result for Task ID %s: %v", job.JobID, err)
    }

    log.Printf("|AGENT TASK|-> Successfully sent result for Task ID %s.", job.JobID)

}
```

Now of course we need to create this new SendResult()

agent/agent.go

And now of course we just need SendResult() I'll place this inside of agent.go

```go
func (agent *Agent) SendResult(resultData []byte) error {

	targetURL := fmt.Sprintf("https://%s/results", agent.serverAddr)

	log.Printf("|RETURN RESULTS|-> Sending %d bytes of results via POST to %s", len(resultData), targetURL)

	// CREATE THE HTTP POST REQUEST
	req, err := http.NewRequest(http.MethodPost, targetURL, bytes.NewReader(resultData))
	if err != nil {
		log.Printf("|❗ ERR SendResult| Failed to create results request: %v", err)
		return fmt.Errorf("failed to create http results request: %w", err)
	}

	// SET THE HEADERS
	req.Header.Set("Content-Type", "application/json")

	// EXECUTE THE REQUEST
	resp, err := agent.client.Do(req)
	if err != nil {
		log.Printf("|❗ ERR | Results POST request failed: %v", err)
		return fmt.Errorf("http results post request failed: %w", err)
	}
	defer resp.Body.Close() // Close body even if we don't read it, to release resources

	log.Printf("💥 SUCCESSFULLY SENT FINAL RESULTS BACK TO SERVER.")
	return nil
}
```

Right now we can't test

```go
func registerCommands(agent *Agent) {
	// agent.commandOrchestrators["shellcode"] = (*Agent).orchestrateShellcode
```

```
    // Register other commands here in the future
}
```

- We have to uncomment to be able to perform lookup
- But, we cannot uncomment till we have orchestrateShellcode

So for now now let's move on with our orchestrator

COPY CODE TO FOLDER