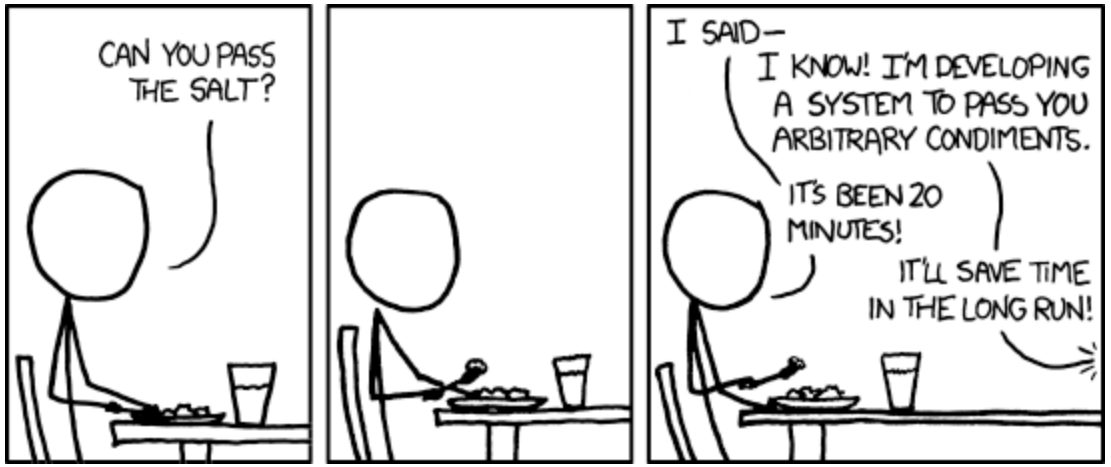
Incremental automation & internal team CLIs

Farid Nouri Neshat

me?

- Senior vibe coder
- Software engineer at heart
- Officially AWS platform engineer specalized in security.
- Unnecessarily automates processes!
- Horrible Guitar Player!

Obligatory XKCD: The General Problem?



Source: xkcd.com/974

Toil?

> long strenuous fatiguing labor

Source: merriam-webster.com dictionary

Toil?

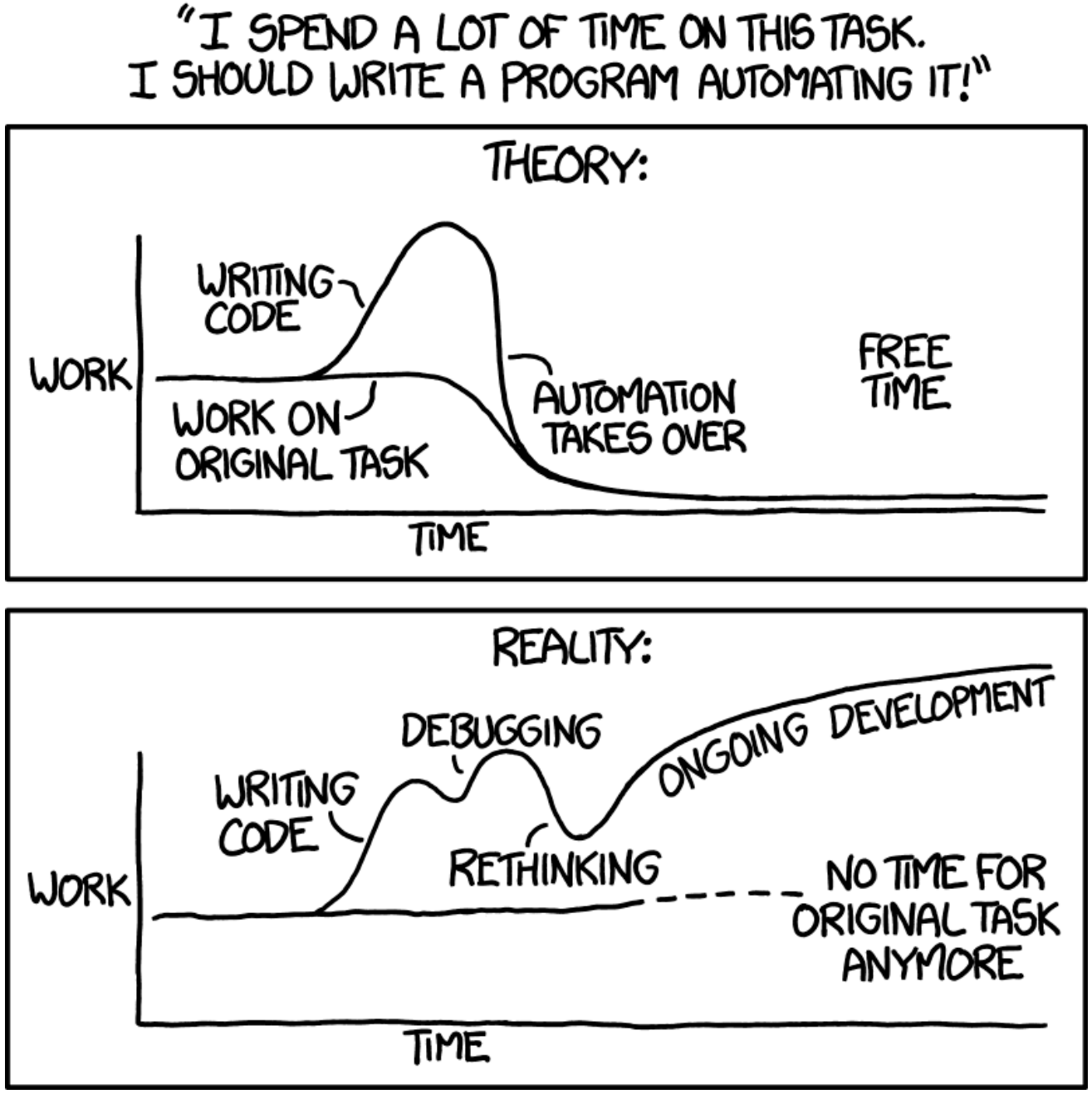
SRE books definition:

- Manual
- Repetitive
- Automatable
- Nontactical/reactive - distracting you from real work
- No enduring value
- O(n) with service growth - Have to do more of it as the source grows

Source: "Site Reliability Engineering" by Betsy Beyer, Chris Jones, Jennifer Petoff and Niall Richard Murphy. Copyright © 2016 Google, Inc.

> 'Automating' comes from the roots 'auto-' meaning 'self-', and 'mating', meaning 'screwing'.

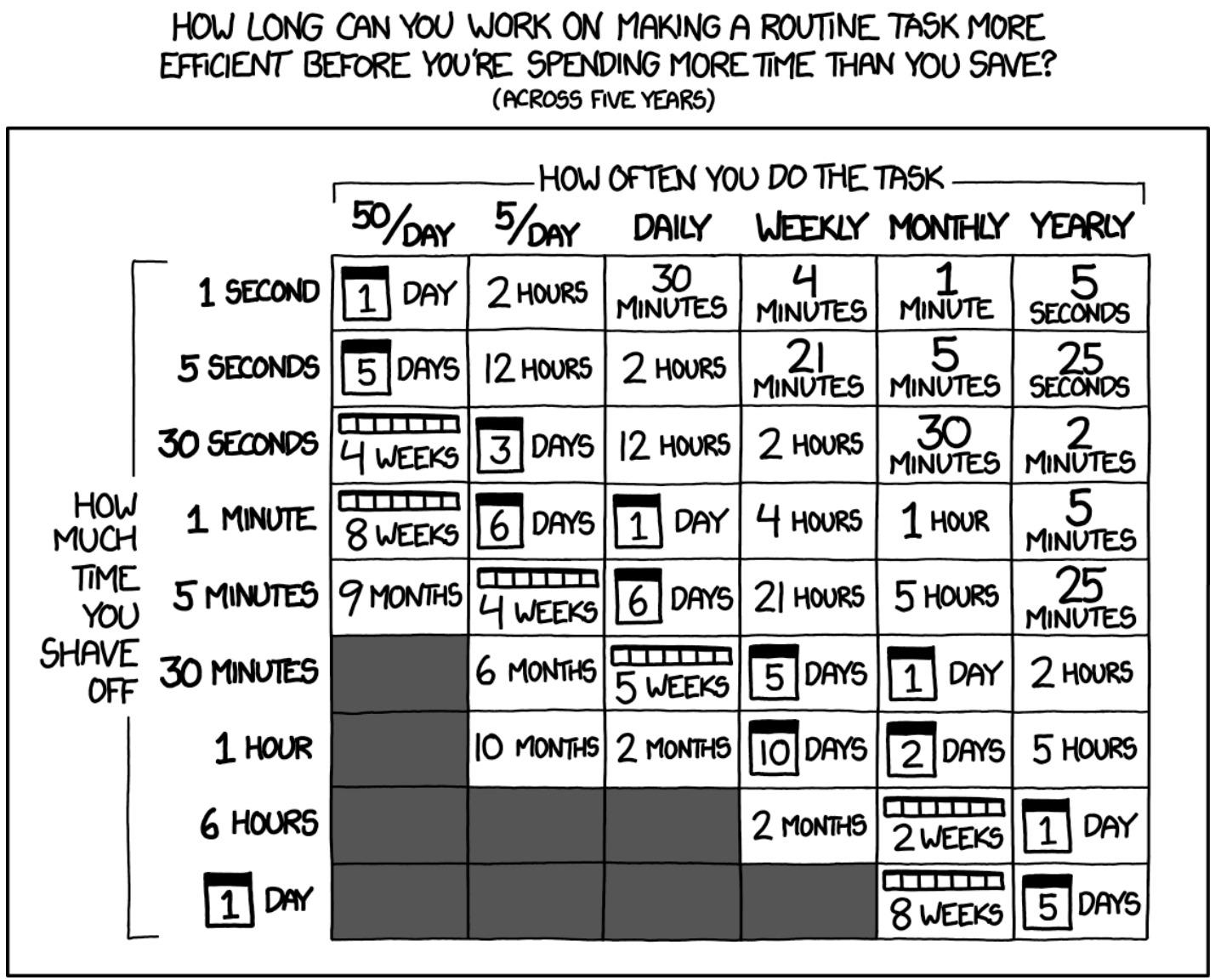
Source: xkcd.com/1319



Repetitive manual tasks?

- Something that needs to be manually set in database
- Operations outside of code
- Gathering data for troubleshooting
- Adding scafolding for a new feature
- Creating a dev container
- Creating a new AWS account
- Setting up development environment

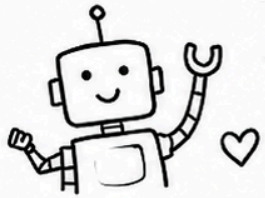
Source: xkcd.com/1205



Toil tax: Iterative automation

1. The very first time: No automation, just do it manually!
2. Start with documenting the steps. Make a runbook!
3. Make them into a do-nothing script!
4. Everytime you do the task. Take the least amount of time to improve the automation.
5. The next time you'll see if you managed to improve it or not! Improve it again!

   ...
6. ... You have fully automated it!



Is This Micro-Automation Worth It?

Cells show how much time you can spend improving the process EACH TIME you do it and still hit 10% cumulative ROI after 1 year.

how much time you shave off each run ↓	how often you do the task			
	daily	weekly	monthly	quarterly
1 second	2.8 min	23.2 sec	5 sec	1.4 sec
5 seconds	13.8 min	1.9 min	25 sec	6.8 sec
15 seconds	41.4 min	5.8 min	1.25 min	20.5 sec
30 seconds	1.4 hr	11.6 min	2.5 min	40.9 sec
1 minute	2.8 hr	23.2 min	5 min	1.4 min
5 minutes	13.8 hr	1.9 hr	25 min	6.8 min

Formula: allowed improvement time per run = shaved time × (annual frequency - 1) / 2.2

In a rush?

Would it have helped if the task took less time or less error prone?

More reason to automate afterwards!

Hate driven development!

Do you have hate the task?

Use your hate to automate!

Do-nothing script!

```
def step_1_query_flow_logs():
    print("1. Query VPC Flow Logs.")
    input("Press Enter when done...")


def step_2_lookup_interface():
    print("2. Look up the network interface.")
    input("Press Enter when done...")
```

After a while a bunch of random scripts around!

- Put them all in a repository
- Slowly document them
- Make them all with same conventions
- Reuse components in their own libraries
- Put them in a nice CLI framework
- Get your team and agents use it!

Typer. The nice CLI framework!

```python
import typer
app = typer.Typer()

@app.command()
def hello(name: str):
    print(f"Hello {name}")

if __name__ == "__main__":
    app()
```

```
$ python main.py hello --help

 Usage: main.py hello [OPTIONS] NAME

╭─ Arguments ──────────────────────────────────────────────────╮
│ *    name       TEXT  [default: None] [required]            │
╰──────────────────────────────────────────────────────────────╯
╭─ Options ────────────────────────────────────────────────────╮
│ --help          Show this message and exit.                  │
╰──────────────────────────────────────────────────────────────╯
```

How about AI?

- The agents can use your CLI
- Use the docstring & help functions for orientation and documentation
- Abstract away common high token tasks
- Can be comebined with skills
- You can build safety features to restrict them
- Allow for json output to do further composition

Safety with agents (& humans)

- Keep credentials in system keychain
- Make commands dry-run default
- Build-in imdepotentency. Running "create-aws-account --ticket JIRA-1231" twice shouldn't create two accounts
- Have feature flags that can be set config to disable dangerous features!

Demo: AWS cost & network troubleshooting

- Cost spike: who moved the bytes?
- Blocked traffic: what cannot connect?
- For both: Check VPC Flow Logs, network interfaces & DNS logs

Two runbooks first

Cost spike:

1. Query top bytes

```
SELECT interface_id, srcaddr, dstaddr, dstport, bytes
FROM flow_logs
ORDER BY bytes DESC
LIMIT 5
```

2. Look up network interface info
3. Find hostname for destination IP in DNS query logs

Blocked traffic:

1. Query blocked flows

```
SELECT interface_id, srcaddr, dstaddr, dstport, action
FROM flow_logs
WHERE action = 'REJECT'
LIMIT 5
```

2. Look up network interface info
3. Find hostname for destination IP in DNS query logs

Do-nothing scripts

```
def step_1_query_flow_logs():
    print("1. Query VPC Flow Logs for top bytes.")
    input("Press Enter when done...")


def main():
    step_1_query_flow_logs()
    step_2_lookup_interface()
    step_3_check_dns()
```

Automate one step

```
FLOW_LOGS_QUERY = """
SELECT interface_id, srcaddr, dstaddr, bytes
FROM flow_logs
ORDER BY bytes DESC
LIMIT 5
"""

def step_1_query_flow_logs():
    now = datetime.now(UTC)

    started = logs.start_query(
        logGroupName=FLOW_LOG_GROUP,
        startTime=int((now - timedelta(days=days)).timestamp()),
        endTime=int(now.timestamp()),
        queryString=FLOW_LOGS_QUERY,
    )
    pprint(logs.get_query_results(queryId=started["queryId"]))
```

Move code into a library so it can be reused!

```
FLOW_LOGS_QUERY = """
SELECT interface_id, srcaddr, dstaddr, dstport, action, bytes
FROM flow_logs
ORDER BY bytes DESC
LIMIT 5
"""
def step_1_query_flow_logs():
    pprint(core.query_flow_logs(FLOW_LOGS_QUERY))
```

Merge into a CLI

```
app = typer.Typer()

@app.command()
def analyze_cost(days: int = 7):
    payload = core.analyze_costs(days)
    print_findings(payload["findings"], "bytes")

@app.command()
def analyze_blocked(days: int = 7):
    payload = core.analyze_blocked(days)
    print_findings(payload["findings"], "blocked")
```

Allow different outputs

```
def show_flow_logs(payload, output_format: str) -> None:
    if output_format == "json":
        typer.echo(json.dumps(payload, indent=2, sort_keys=True))
        return

    table = Table(title="Findings")
    table.add_column("Source IP")
    table.add_column("Name")
    table.add_column("Destination Domain")

    for item in payload:
        table.add_row(
            item["source_ip"],
            item["interface_name"],
            item["destination_domain"],
        )

    console.print(table)

show_flow_logs(core.analyze_costs(days), output_format)
```

Questions?