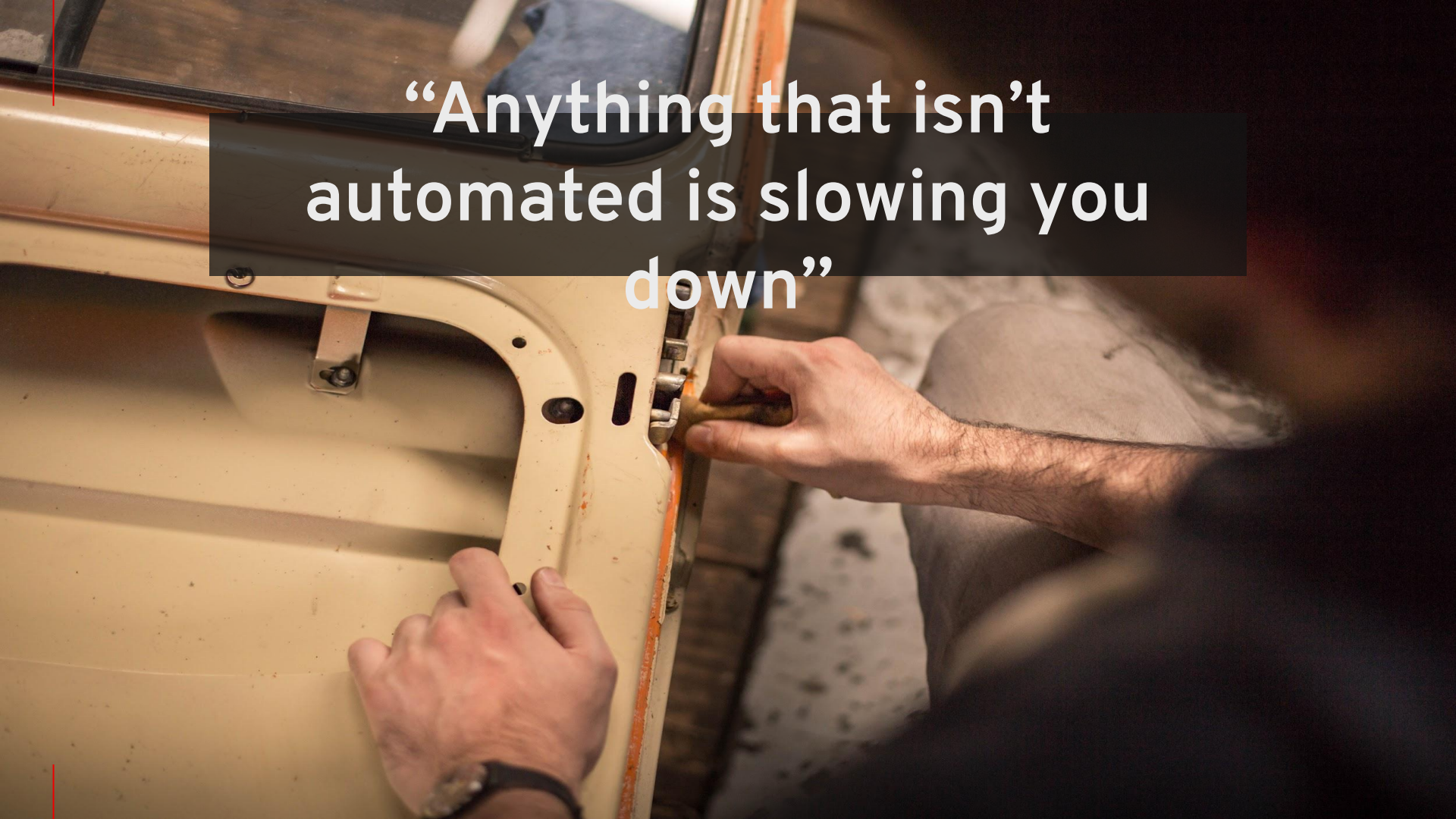
Any application in any system must be **installed**, **configured**, **managed** and **upgraded** over time

Patching is critical to security


“Anything that isn’t automated is slowing you down”

Operator Maturity Model

Phase I	Phase II	Phase III	Phase IV	Phase V
Basic Install	Seamless Upgrades	Full Lifecycle	Deep Insights	Auto Pilot
Automated application provisioning and configuration management	Patch and minor version upgrades supported	App lifecycle, storage lifecycle (backup, failure recovery)	Metrics, alerts, log processing and workload analysis	Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning

Site Reliability Engineering (SRE)

- O’Reilly “SRE Book” (Beyer et al)
- Carla Geisser (al) paraphrased: ~“*Human intervention… is a bug*”
- SREs write code to fix those bugs
- SREs write software to run other software
- SREs write Kubernetes Operators

Level 1

Installation - Deployment

- Can you set operand configuration in the CR?
- Do CR changes cause non-disruptive updates to the Operand?
- Does CR status show what has and hasn’t been applied?

Level 2

Upgrades

- Can the Operator upgrade its Operand?
- Without disruption?
- Does CR status show what has and hasn’t been upgraded?

Level 3

Full Lifecycle Management

- Can your Operator back up its Operand?
- Can your Operator restore from a previous Operand backup?
- Ready/Live probes? Active monitoring of basic execution state?
- CPU and other requests and limits set for Operand?

Level 4

Deep Insights

- Does the Operator expose metrics about its own health?
- Metrics and alerts for the Operand?
- Does CR status show what has and hasn’t been applied?

RED

Rate (aka Traffic) - Errors - Duration (aka Latency)

The RED Method defines three key metrics for every service

- Rate (the number of requests per second)
- Errors (the number of those requests that are failing)
- Duration (the amount of time those requests take)

Level 5

Auto Pilot

- Marine autopilots are reasonable models, especially with rudder position feedback
- Auto scaling, healing, tuning
    - Detect condition from metrics, scale horizontally (Replicas) or vertically (Requests/Limits)
    - Think especially about scaling back down; resource savings
    - Detecting deterioration in Operand(s) (based on Level 4's metrics) and take action to redeploy or reconfigure
- CR Status, custom Events: Clear status and *especially error conditions*

Level 5 (cont.)

Auto Pilot

“Toil Not, Neither Spin” (Kubernetes Operators, Dobies & Wood)

SRE defines “toil” as:

- **Automatable** - your computer would enjoy it!
- Without **enduring value** - needs done but doesn’t change the system
- **Grows linearly** with growth of the system

Operator Maturity Model

Phase I	Phase II	Phase III	Phase IV	Phase V
Basic Install	Seamless Upgrades	Full Lifecycle	Deep Insights	Auto Pilot
Automated application provisioning and configuration management	Patch and minor version upgrades supported	App lifecycle, storage lifecycle (backup, failure recovery)	Metrics, alerts, log processing and workload analysis	Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning

Experiments/Challenges

“...left as an exercise for the reader…”

- SRE stuff: Add metrics awareness and tuning to your Operator
- Other APIs / API representations: `k8fs`?
- K8fs presents Kubernetes API as a synthetic file hierarchy
- `% cp manifest.yaml /mnt/k8s/ns/default/deployments/`
- `% echo 3 >/mnt/k8s/ns/default/deployments/myapp/replicas`

Resources

https://operatorframework.io

https://operatorhub.io

https://learn.openshift.com/operatorframework/

http://bit.ly/kubernetes-operators