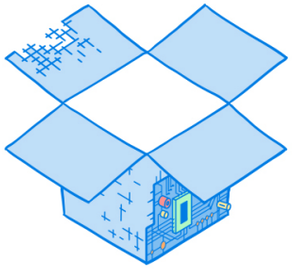
AUTOMATED ERROR HANDLING AT DROPBOX



Maxim Bublis, Károly Nagy

CS3 Workshop, Amsterdam, 2017

About us

Maxim Bublis

SRE
Dropbox

satori

Károly Nagy

MySQL SRE
Dropbox

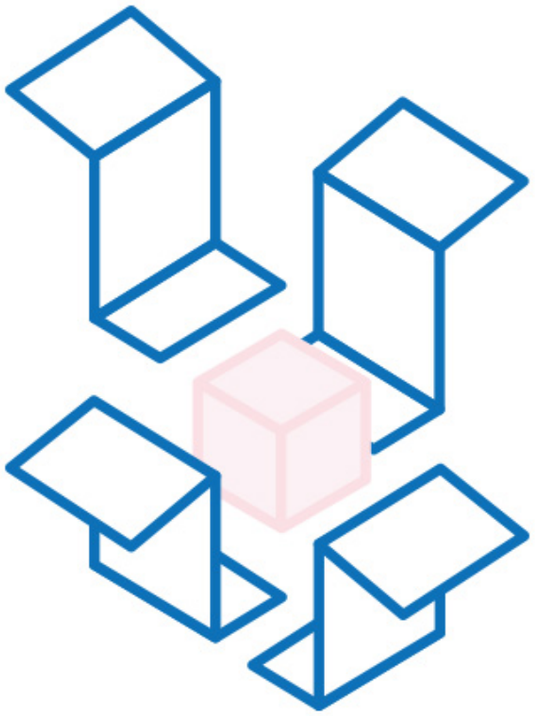
charlesnagy
charlesnagy

What is Dropbox today?

Scale:

- 500 million users
- 200,000 business users
- 1,2 billion files saved every day

What is inside?



Infrastructure

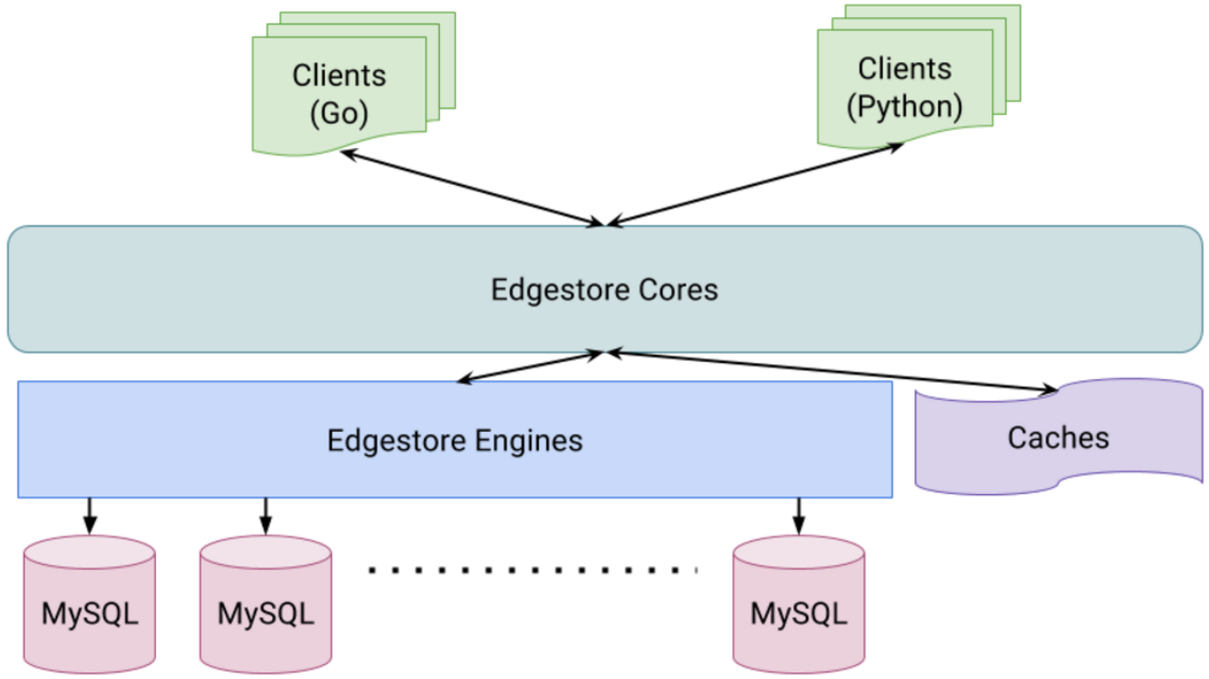
- Application servers
- Metadata storage
- Block storage

Infrastructure

- Application servers
- Metadata storage
- Block storage

Metadata storage

- Sharded MySQL databases
- Edgestore



What is Dropbox again?

- Tens of thousands of servers
- Multiple datacenters
- Multiple regions
- Multiple continents
- Petabytes of metadata
- Exabytes of raw data

Engineering challenges

- Shit happens
- Lots of unknown unknowns
- Way less engineers than hardware


HANG
IN THERE

Different levels of automation

- No automation at all
- Scripting
- Parallel and/or remote execution
- Unattended execution
- Automatic batch job execution
- Self-managing servers

Scripting

Pros:

- Easy
- Good for start
- Predictable

Cons:

- Limited reusability
- Lots of repetition
- Still 1 to 1

Scripting++: parallel and/or remote

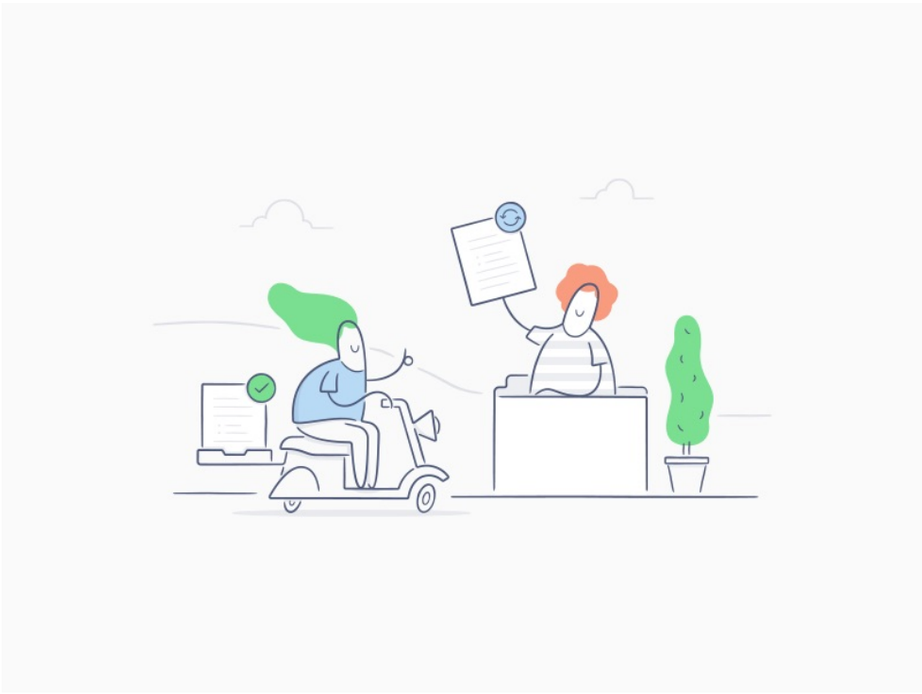
- Run the same script on multiple hosts
- Run it remotely

Examples:

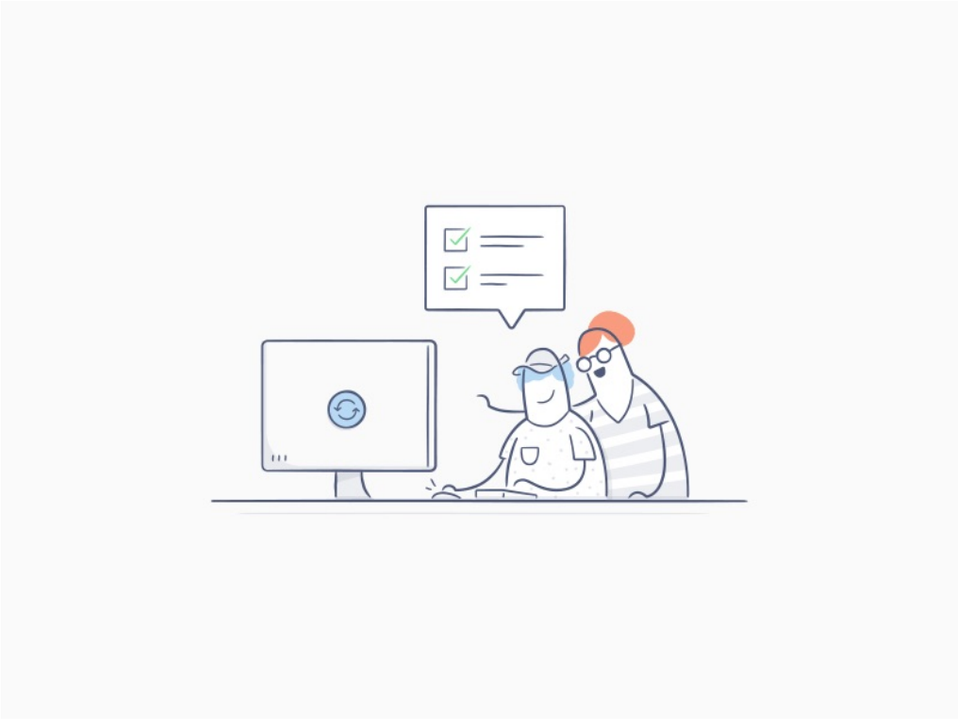
- gsh
- Fabric
- Ansible

Improving further

Unattended execution



Automatic batch execution



Self-managing servers

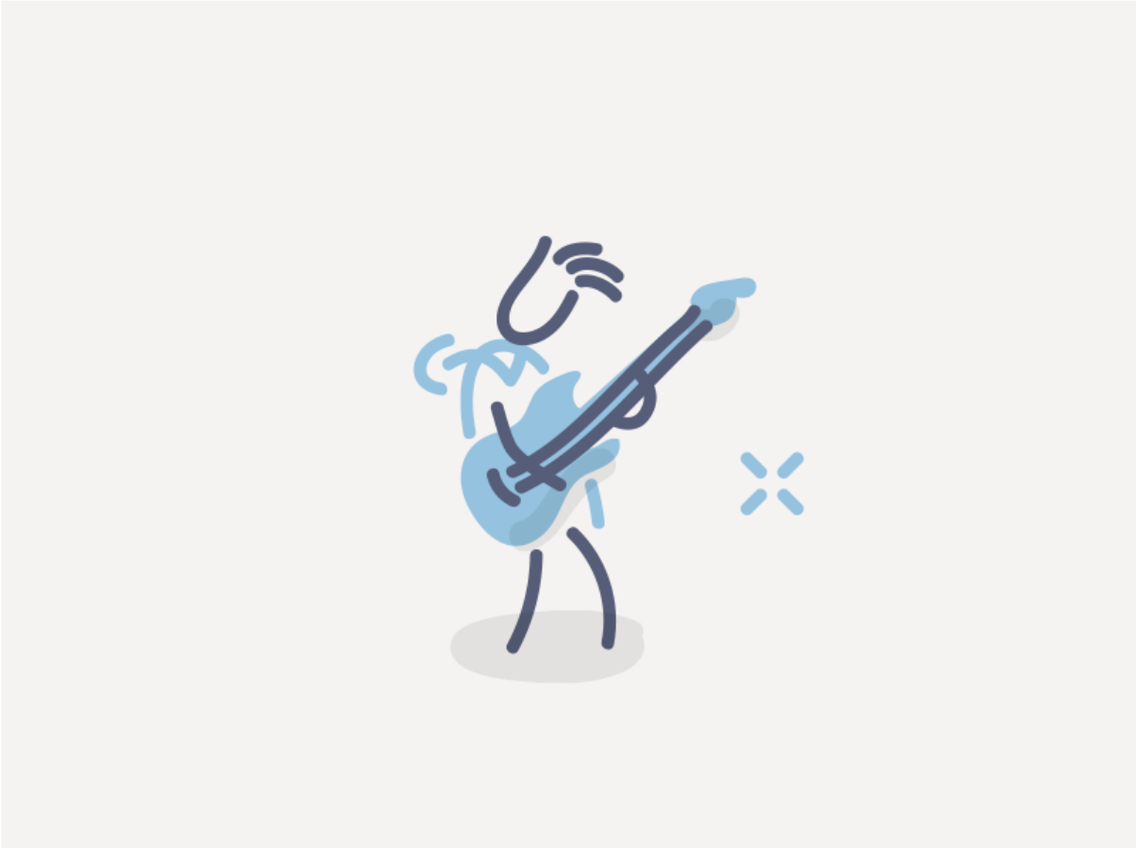
Periodical checks:

- cron

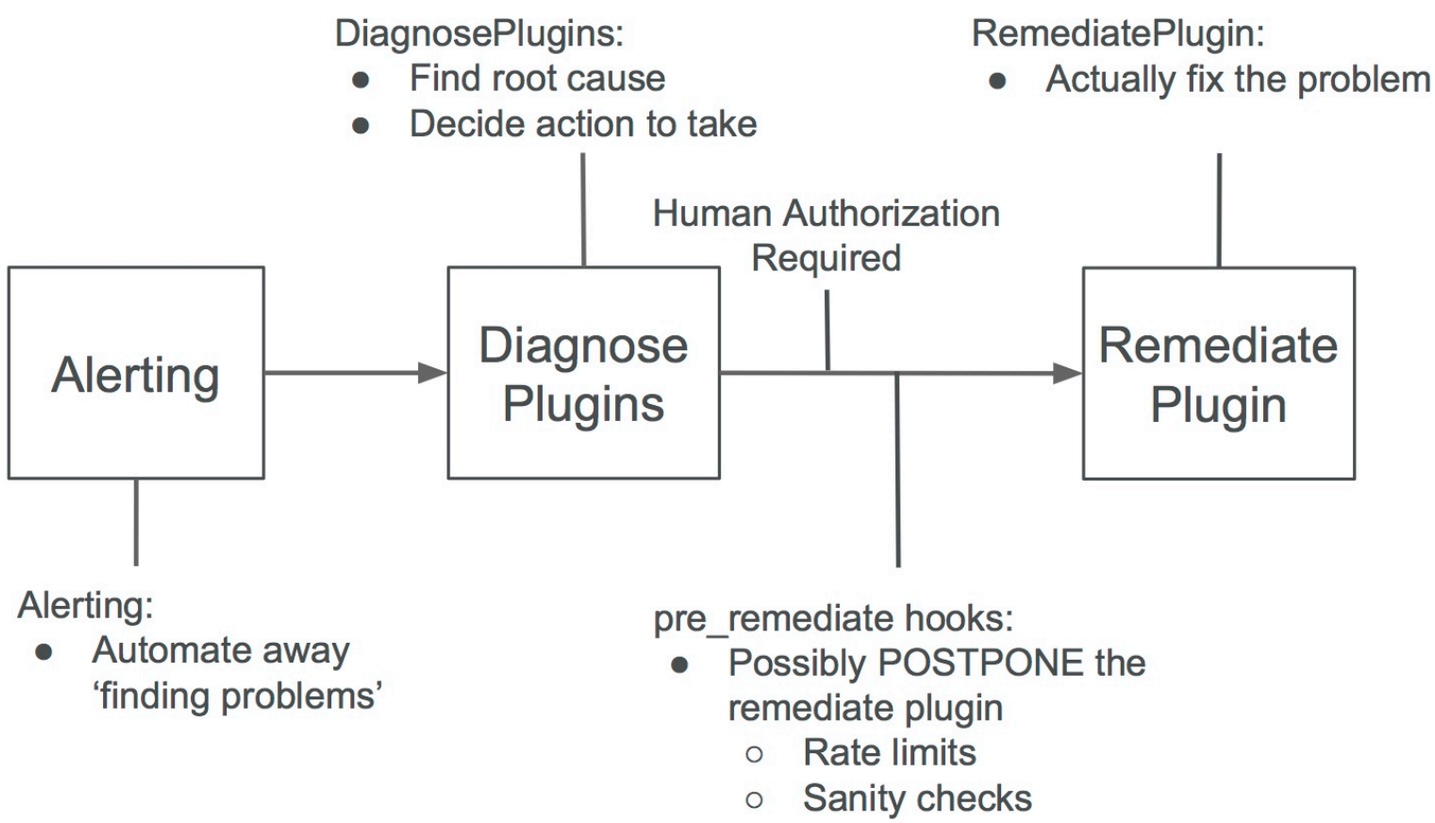
Reactive:

- certain conditions
- events

So you can focus on the important things! ☺️



Automatic remediations: Naoru

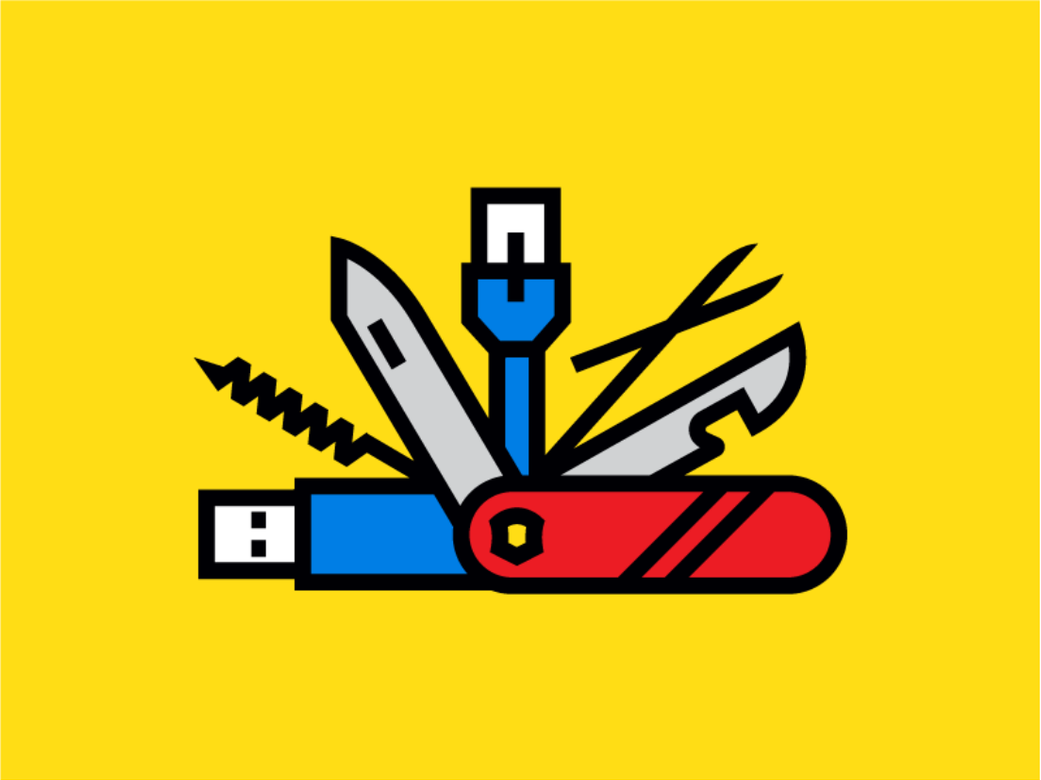


Example: SSHUnreachable alert

1. If SSH actually works → do nothing
2. If repeated reboot history → deallocate
3. Check out-of-band IPMI console
   - If no response → reboot
   - If initramfs shell → deallocate
   - If cpu soft lockup → reboot

Naoru helps a lot

- “Simple” logic
- Alerts are easily handled
  - Process is dead → restart it
  - Disk full → rotate logs
  - Hardware issue → deallocate
- Takes care of great number of issues



There's always more…

- Some tasks are non-trivial
  - Involve multiple hosts
  - Run for hours
  - Require synchronization
  - Require ability to be paused and resumed
- Some tasks must be done immediately, other can wait

We need

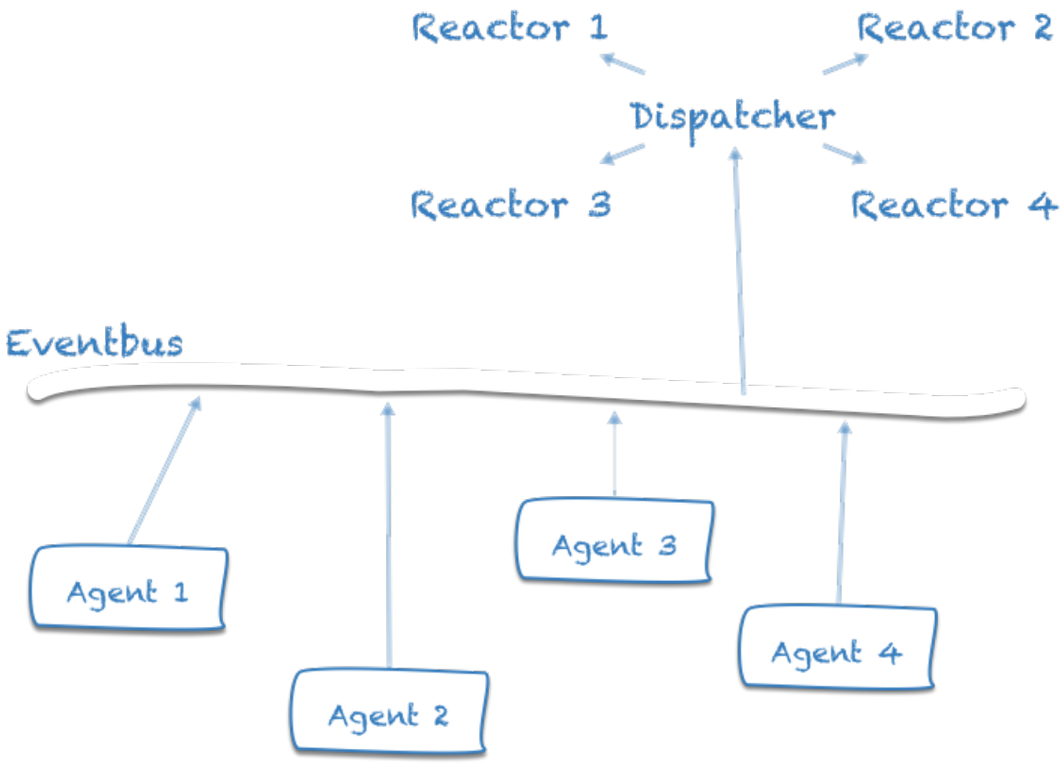
- Smarter scheduler
- Throw different events/signals
- State machines to describe complex workflow
- Prioritization
- Granular tasks: beyond check + run.

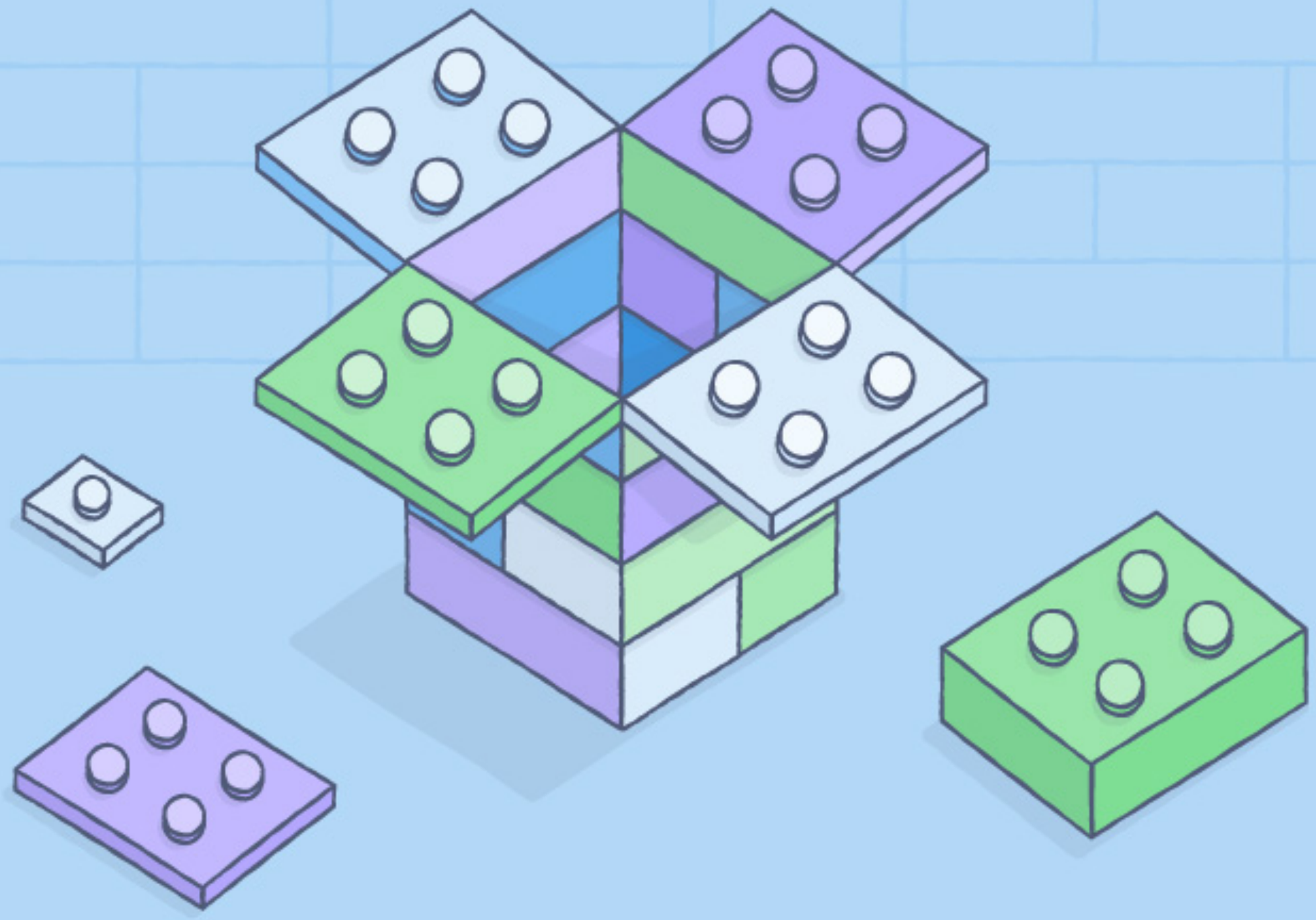
Local agent

Agent
- throws events with some payload

Dispatcher
- takes events from bus
- dispatches them to reactors

Reactor
- subscribes to events
- takes action





Signals / events

Any source of information:

- Nagios alerts
- Service health checks
- Time-series database metrics
- Human action

Event-based FSM

Mathematical model

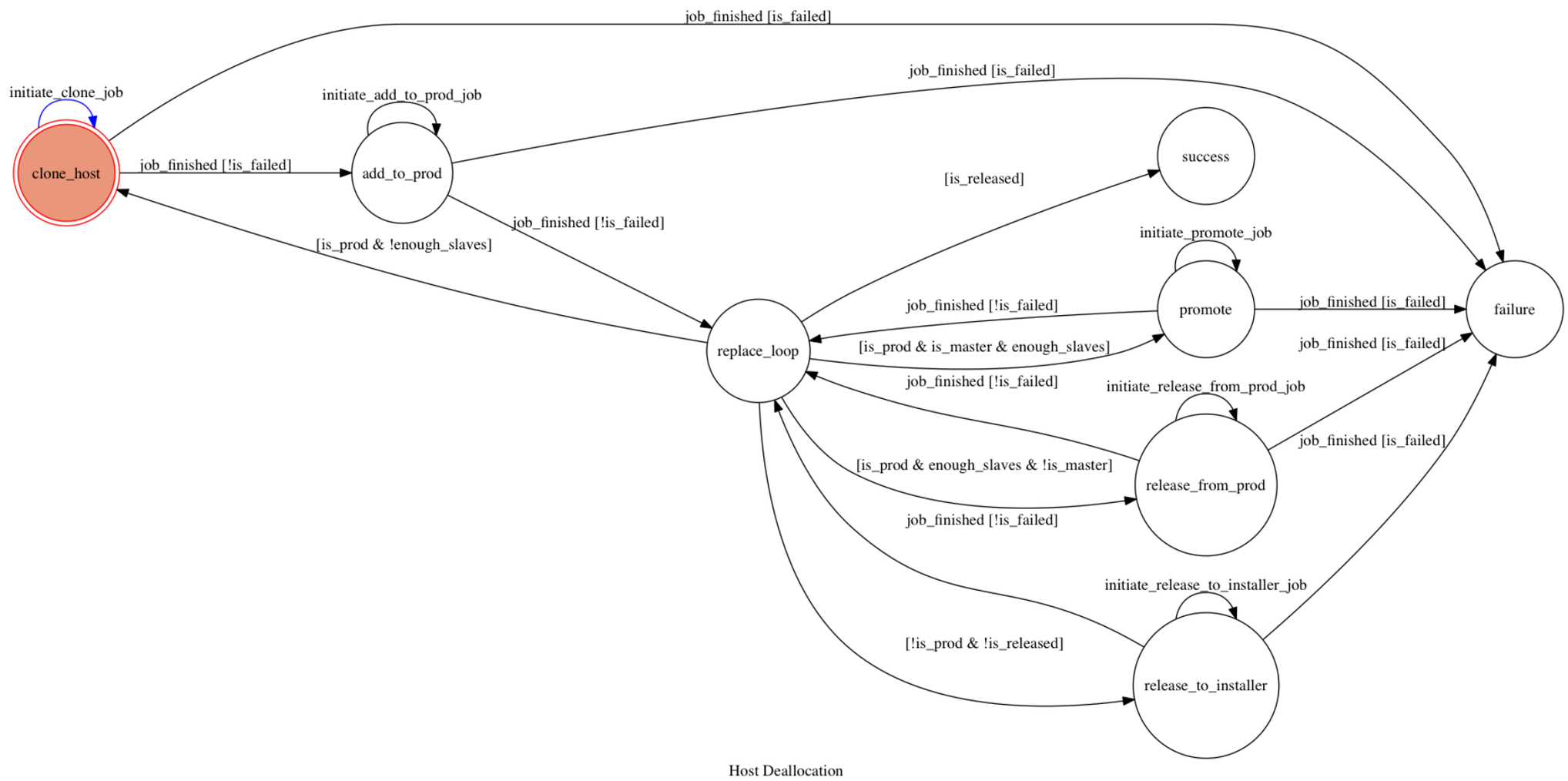
Easy to argue about correctness

Reactive transitions on events

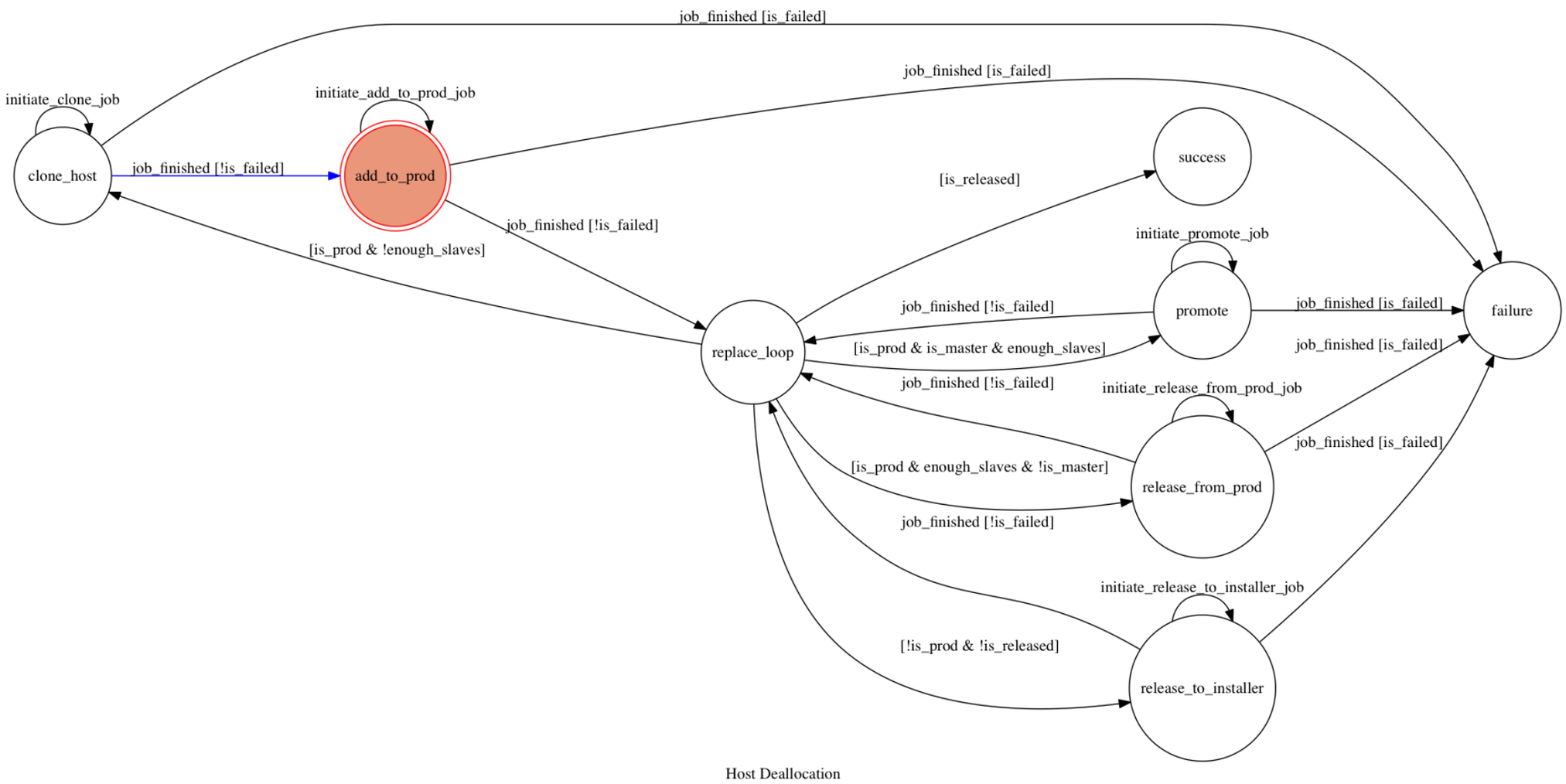
Easy to suspend in between or completely cancel

Easy to represent

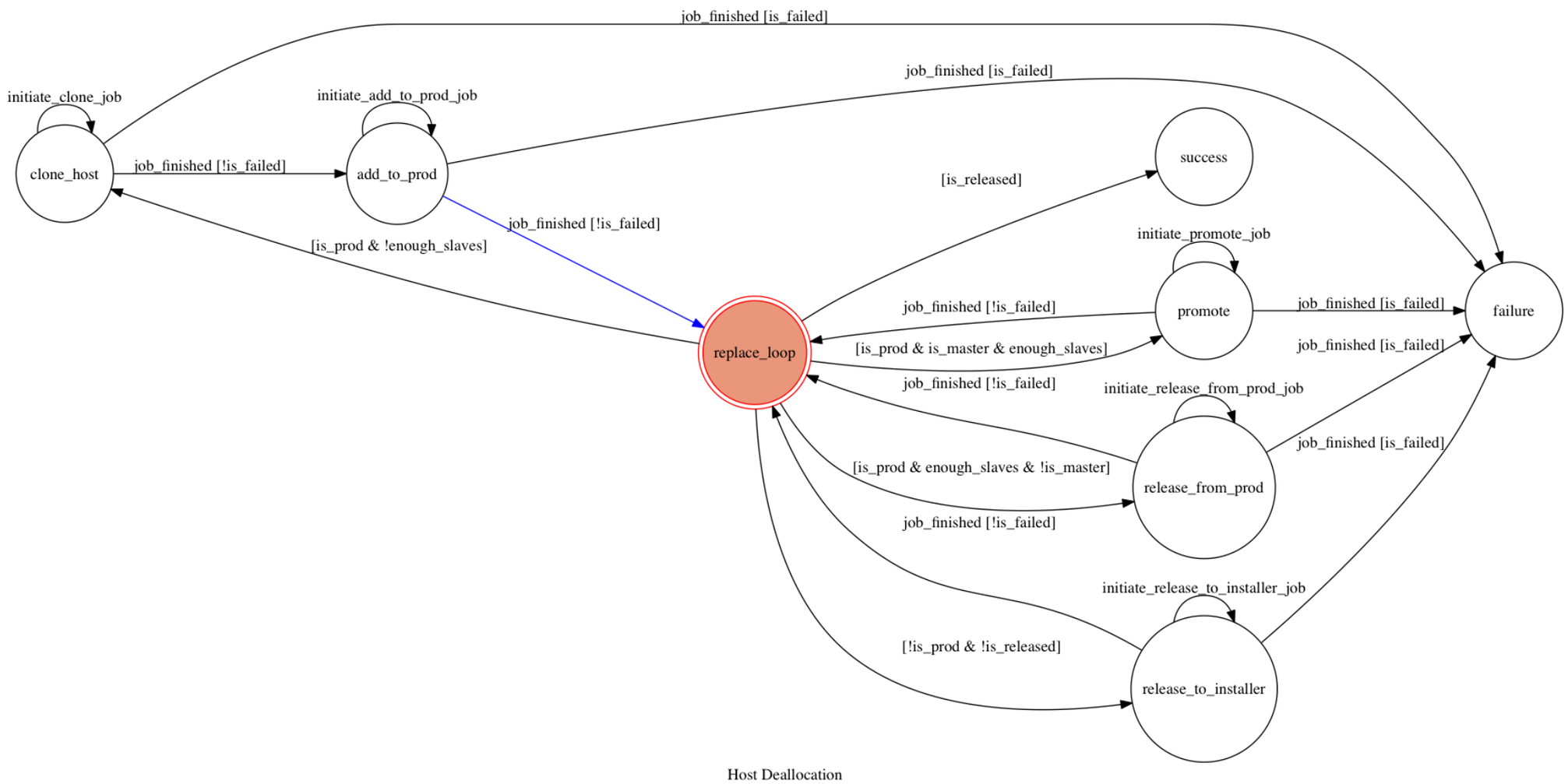
FSM representation



FSM representation



FSM representation

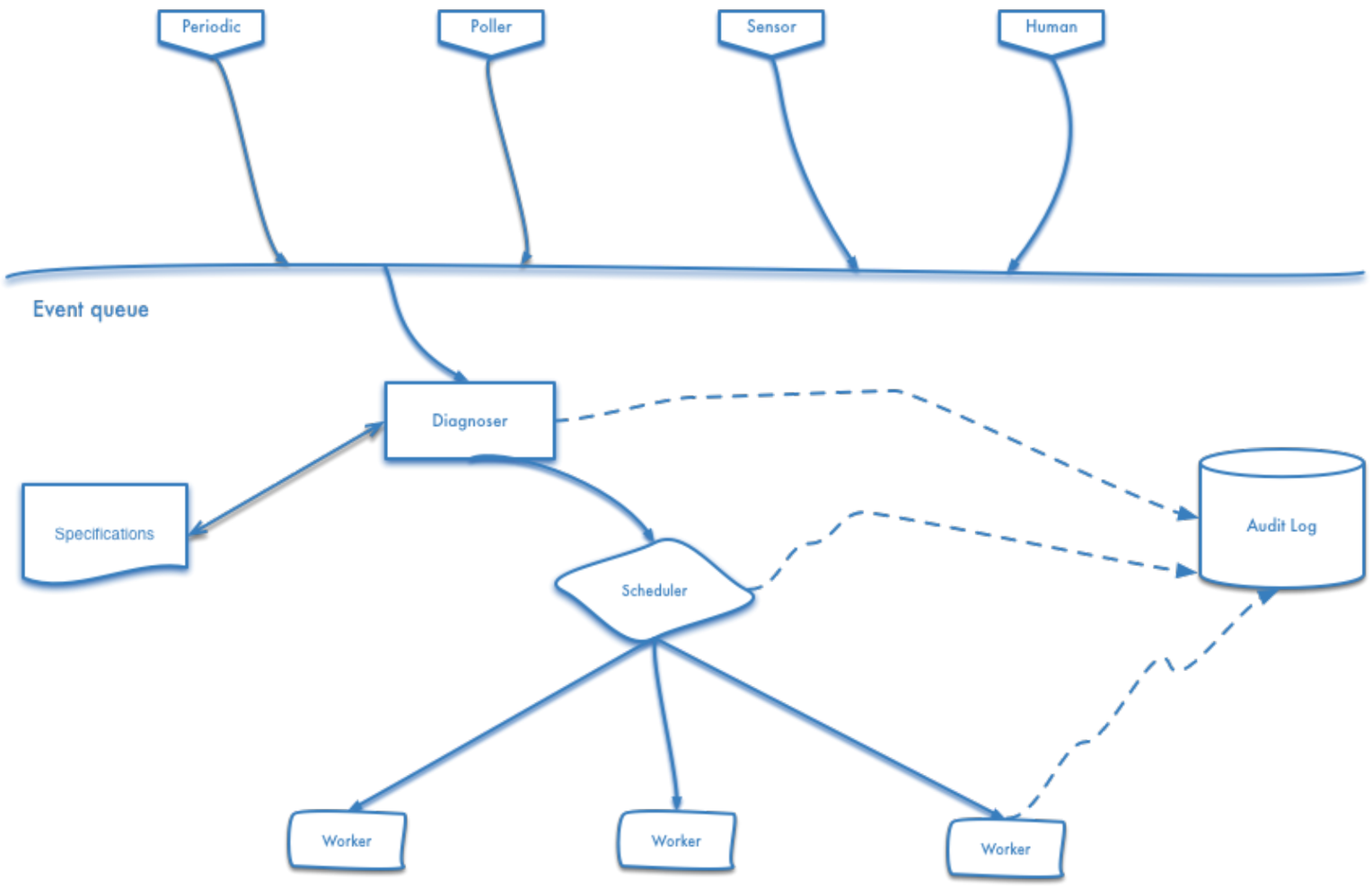


Scheduler

Prioritization

Synchronization

Our solution: Wheelhouse



Example #1: MySQL slave failure

1. Find new machine
2. Make a clone of slave
3. Add new healthy slave to production
4. Remove broken slave from production
5. Send broken slave to reinstall or repair

Example #2: MySQL master failure

Almost the same as #1. But...

1. Promote a healthy slave to be the master

Logic can be reused with hierarchical state machines:

one FSM could be a part of another

Example #3: Kernel upgrade

"Simpler":

1. Allocate new machine
2. Upgrade kernel to required one
3. Add new machine into production
4. Remove machine with old kernel from production

Just maintenance — lower priority

Example #4: Verifying data consistency

1. Clone new slave
2. Stop at defined checkpoint
3. Run data consistency verification
4. Report result
5. Deallocate it

CS3 Workshop, Amsterdam, 2017

Thanks

CS3 Workshop, Amsterdam, 2017