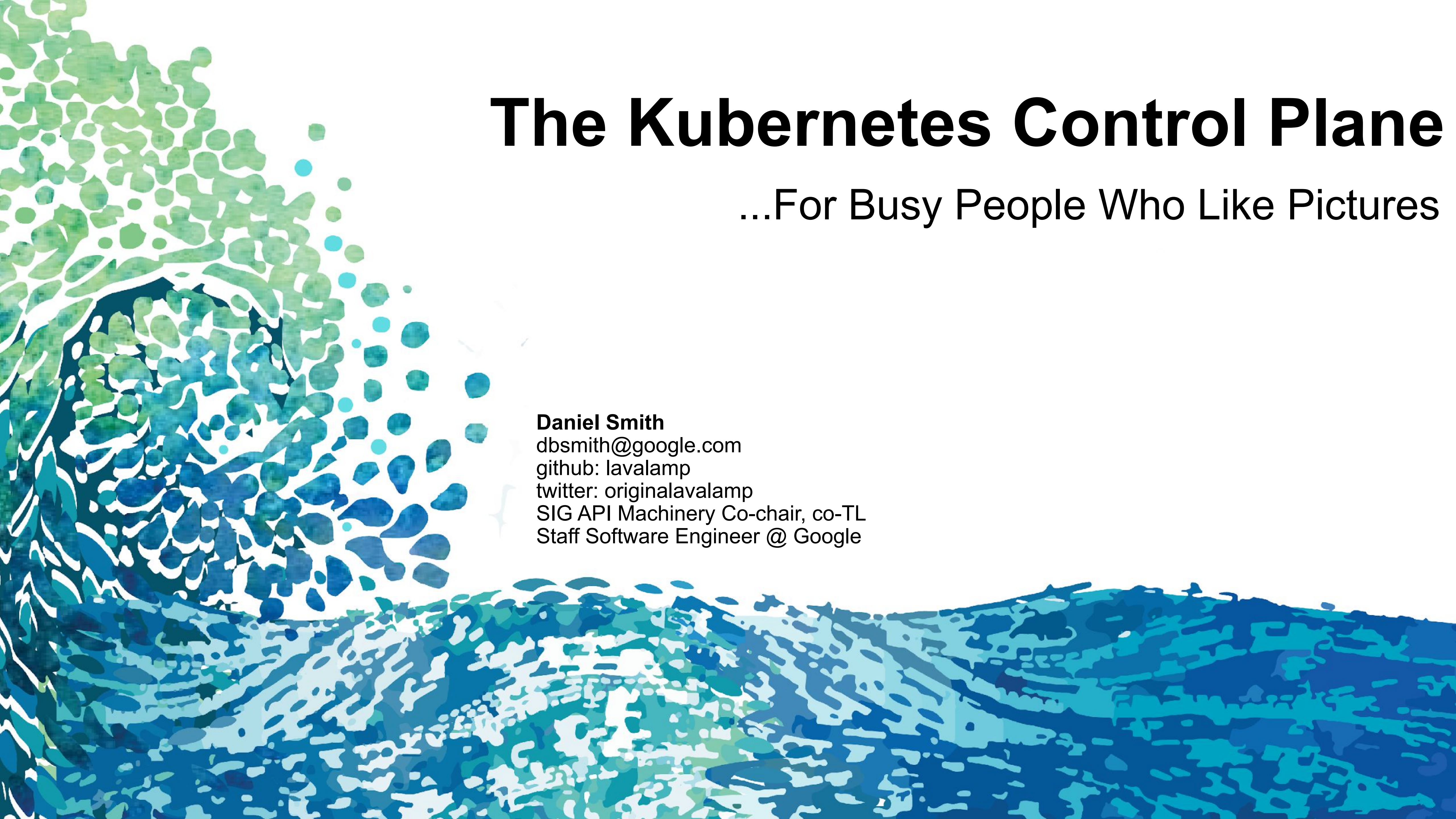


The Kubernetes Control Plane

...For Busy People Who Like Pictures

Daniel Smith
dbsmith@google.com
github: lavalamp
twitter: originalavalamp
SIG API Machinery Co-chair, co-TL
Staff Software Engineer @ Google

DRAFT 2

You’re reading a **DRAFT**! Draft 2 has some pictures!

Final slides coming soon!

These may not make a lot of sense without my commentary! You should come to the talk if you want that :)

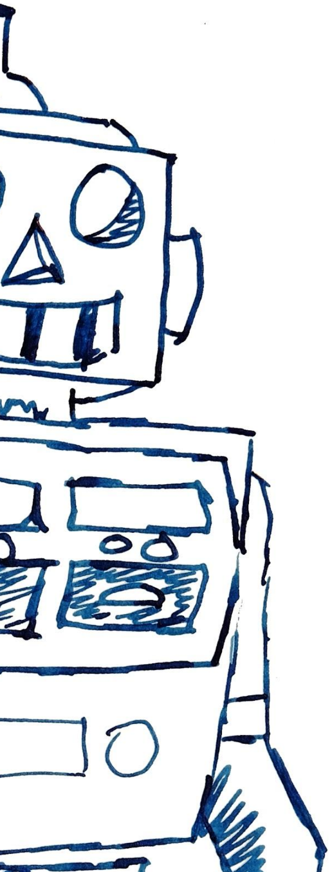
THE KUBERNETES CONTROL PLANE

FOR BUSY PEOPLE

WHO LIKE PICTURES

THE KUBERNETES CONTROL PLANE

FOR BUSY PEOPLE WHO LIKE $_{\wedge}^{\text{BAD}}$ PICTURES

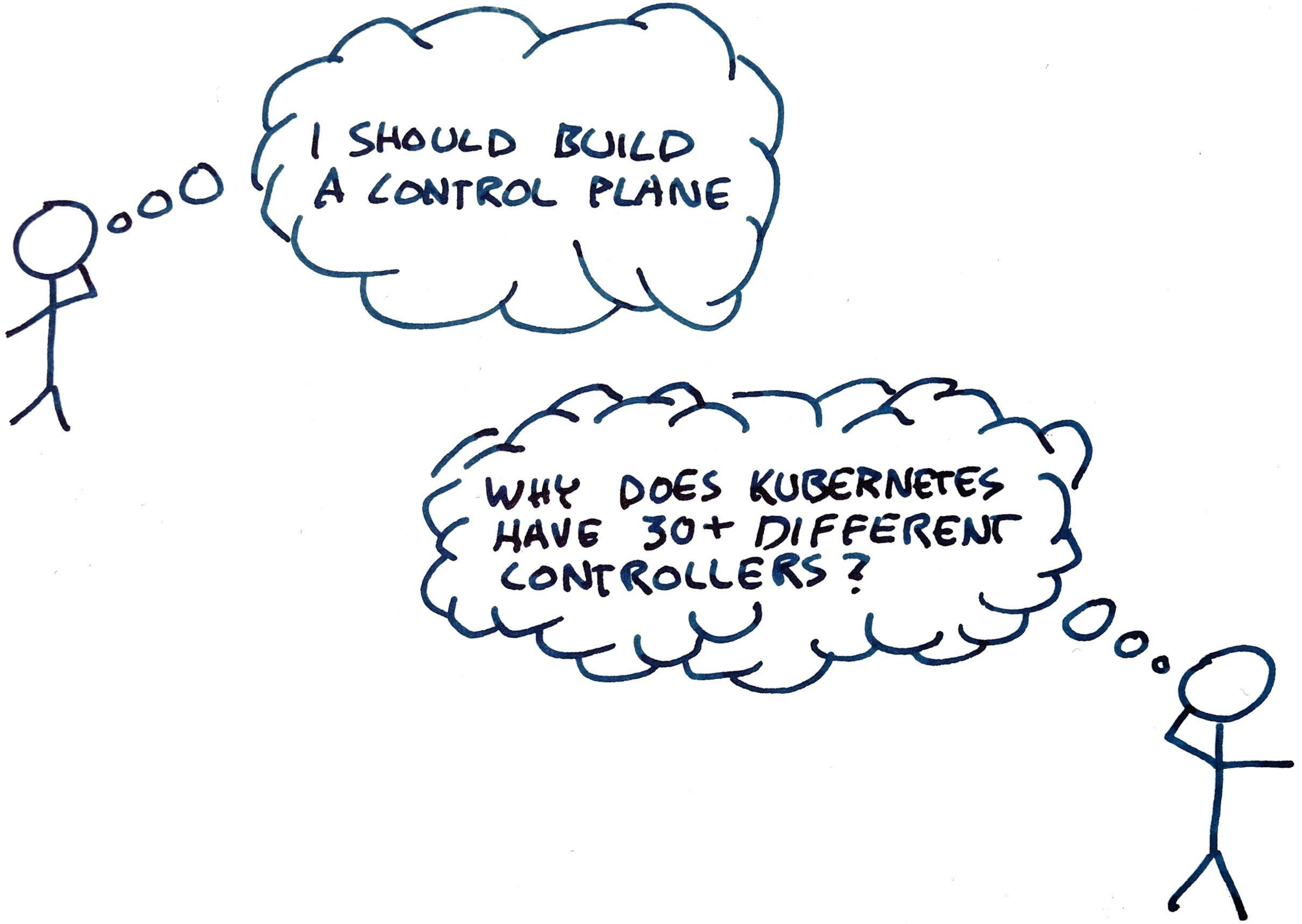


THE KUBERNETES CONTROL PLANE

FOR BUSY PEOPLE WHO LIKE $\overset{\text{BAD}}{\wedge}$ PICTURES

DANIEL SMITH

STAFF SOFTWARE ENGINER — GOOGLE
LAVALAMP — GITHUB
ORIGINALAVALAMP — TWITTER
SIG API MACHINERY
CO-CHAIR * CO-TL


I SHOULD BUILD A CONTROL PLANE
WHY DOES KUBERNETES HAVE 30+ DIFFERENT CONTROLLERS?

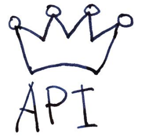
TRILOGY

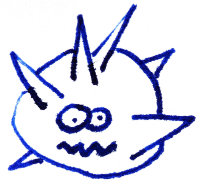
COPENHAGEN 2018: KUBERNETES-STYLE APIs OF THE FUTURE

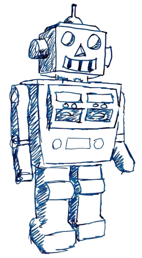
SEATTLE 2018: A VISION FOR API MACHINERY

BARCELONA 2019: THE KUBERNETES CONTROL PLANE FOR BUSY PEOPLE WHO LIKE PICTURES

YOU ARE HERE







THE KUBERNETES API IS ABOUT HUMANS AND MACHINES WORKING TOGETHER.

THE KUBERNETES API IS ABOUT HUMANS AND MACHINES WORKING TOGETHER.

... YOU CAN'T DO THAT WITHOUT SOME MACHINES!!

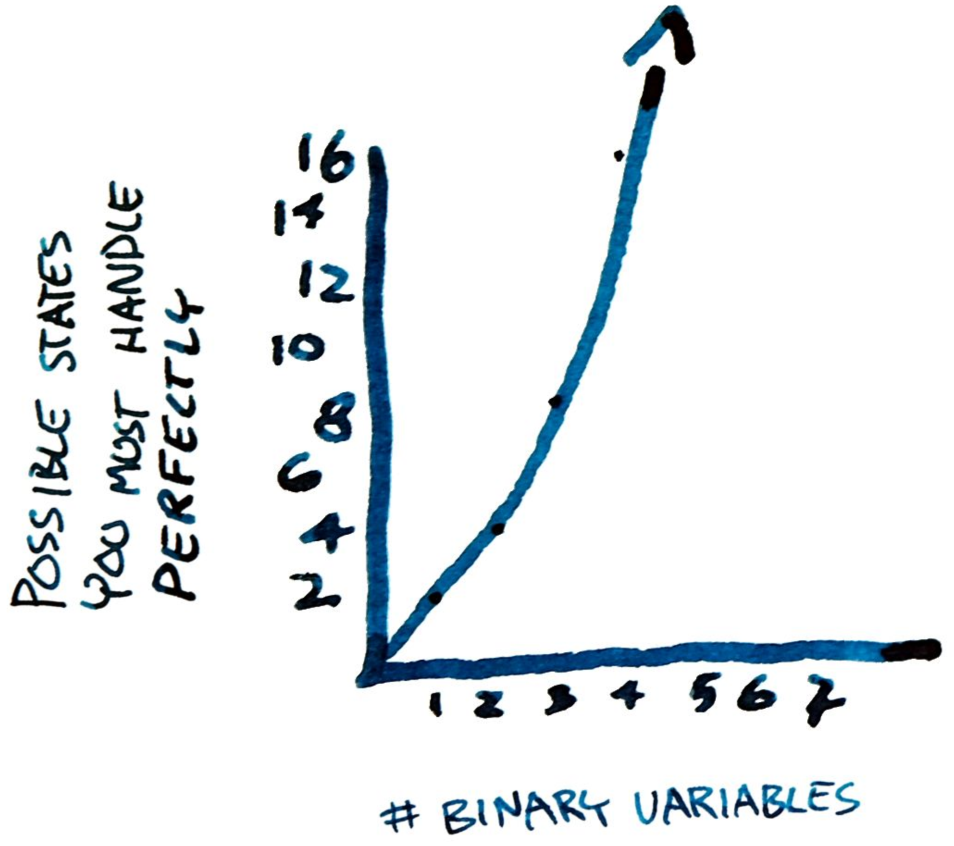
CONTROLLERS: THE IN THE MACHINE

CONTROL THEORY!

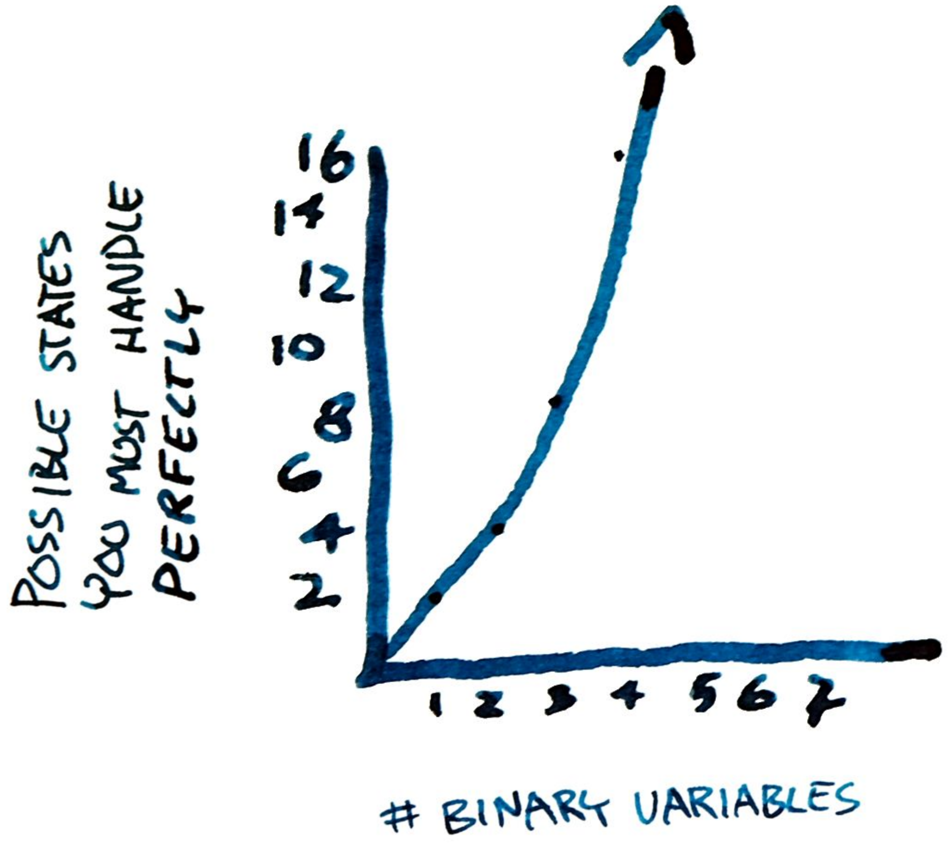
THE AGE-OLD DEBATE: ~~NATURE VS NURTURE~~

STATE MACHINE
VS
CONTROL LOOP

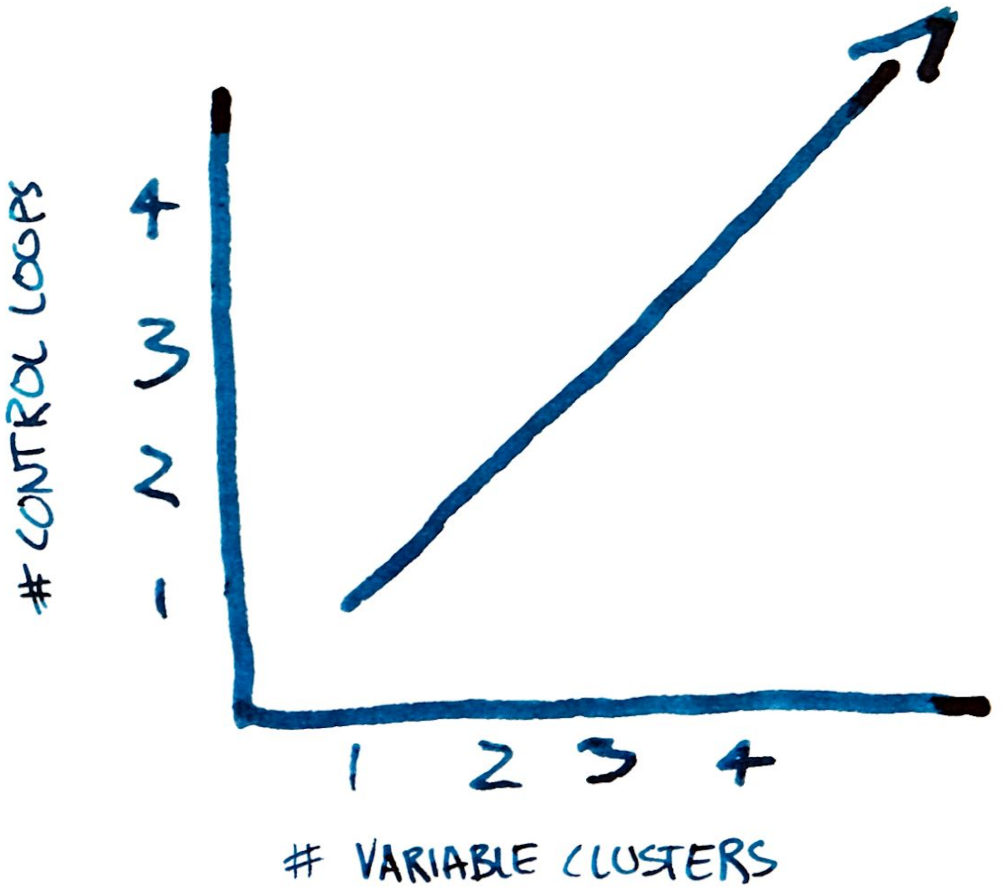
STATE MACHINES



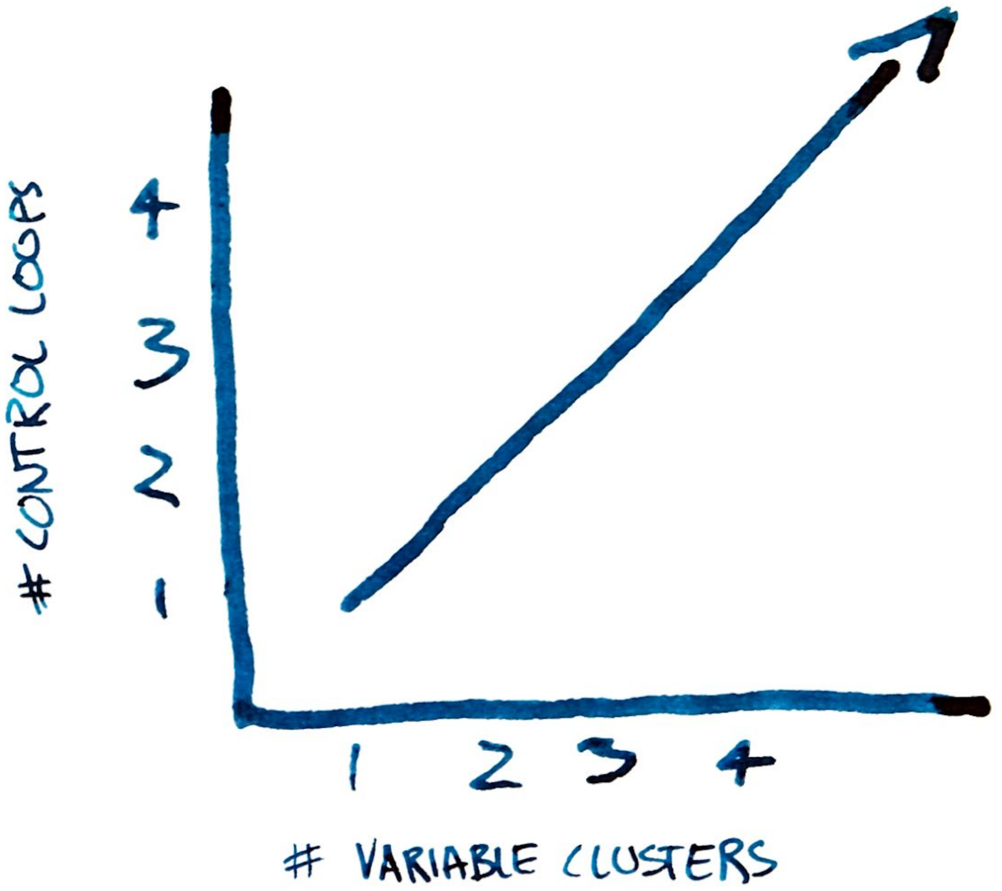
STATE MACHINES



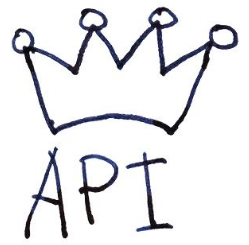
2^N ← TECHNICAL TERM: NOT GOOD ☹

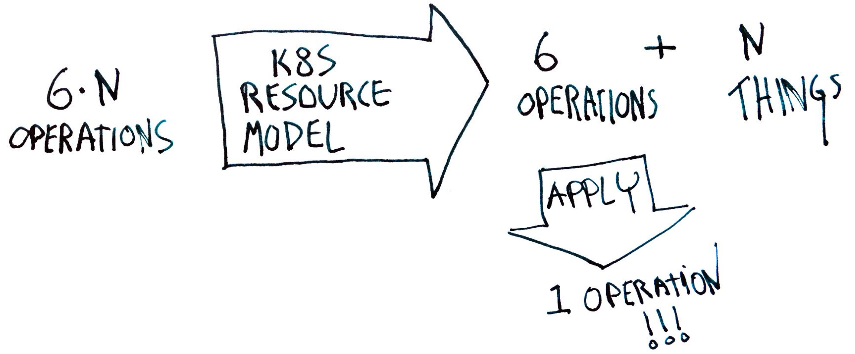


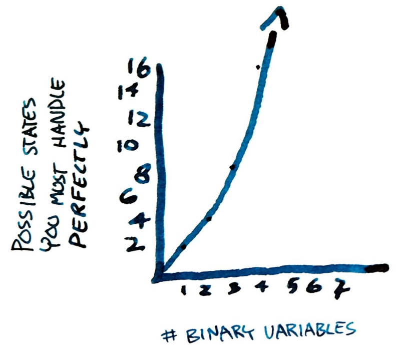
# CONTROL LOOPS
1
2
3
4
1
2
3
4
# VARIABLE CLUSTERS

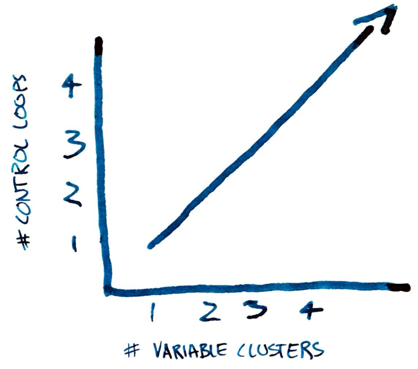


N ← TECHNICAL TERM: PRETTY GOOD ☺









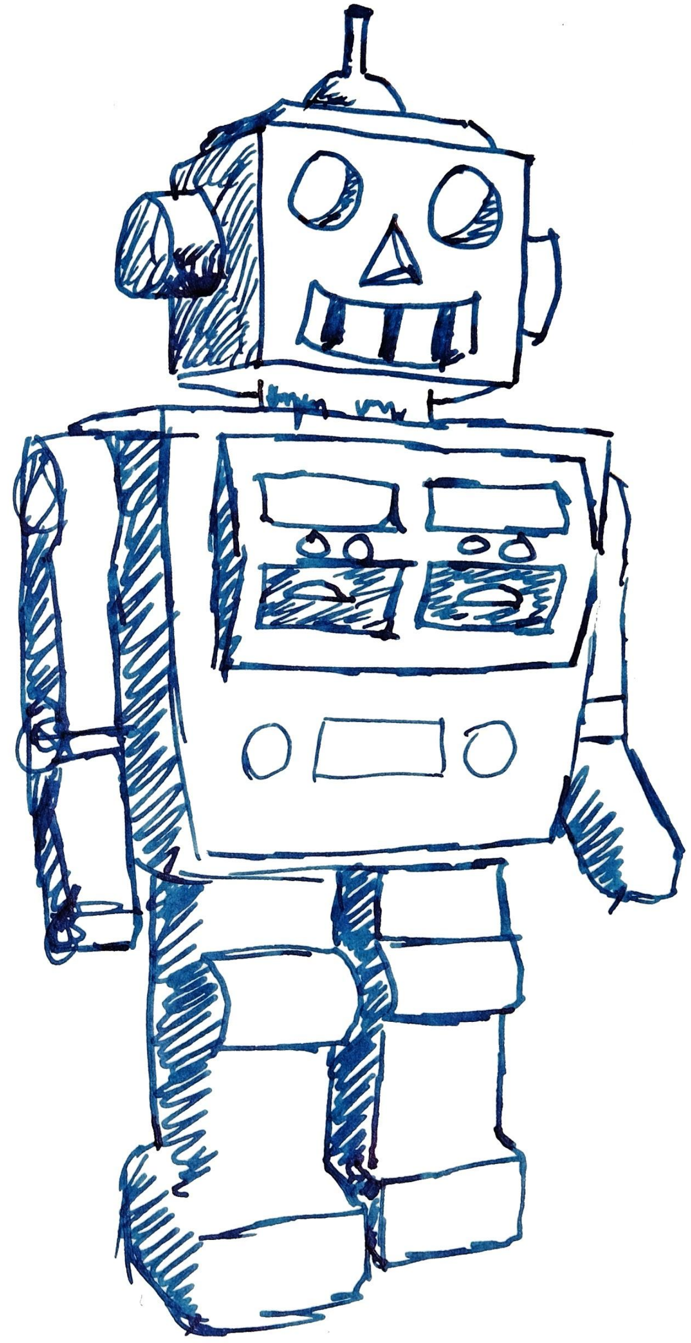
INTEGRATION COMPLEXITY VS IMPLEMENTATION COMPLEXITY

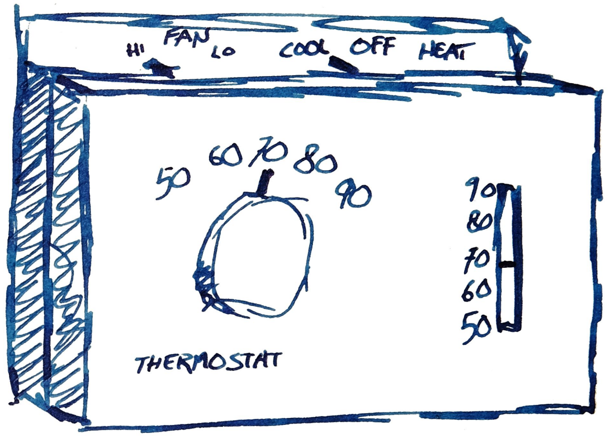
GLOBALLY EASIER
LOCALLY HARDER

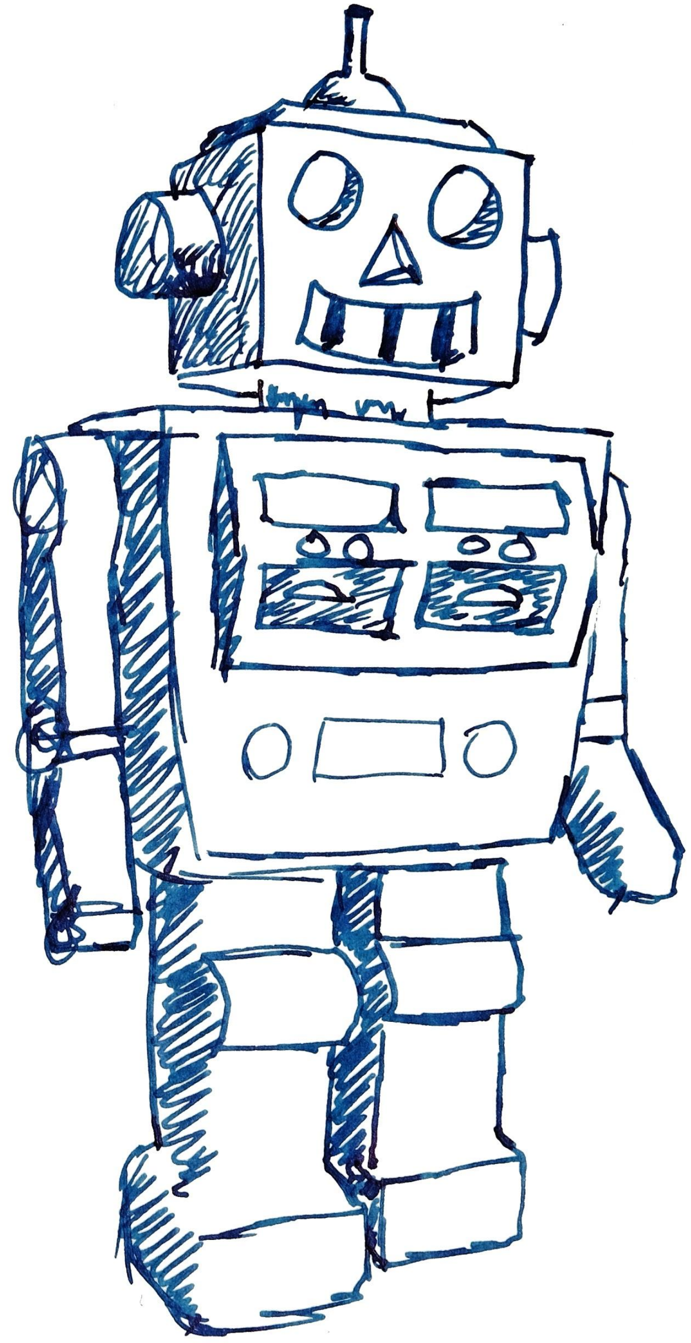
AN IDEAL KRM CONTROLLER

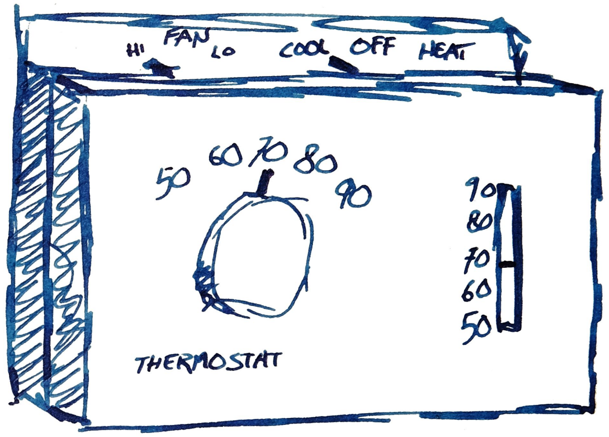
AN **IDEAL KRM CONTROLLER** SHOULD:

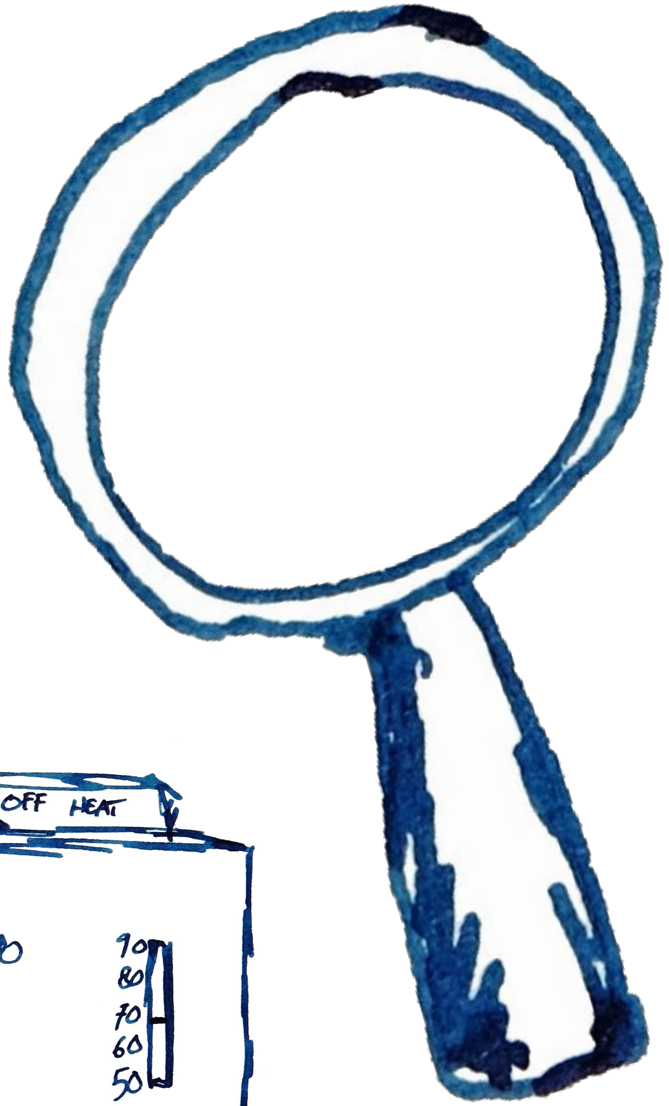
* ONLY DO **ONE** THING
* HAVE AN **INPUT** SOURCE
* HAVE A PLACE TO WRITE **STATUS**
* HAVE AN **OUTPUT** LOCATION
* **ANTICIPATE ITS OWN EFFECTS** ON THE REST OF THE SYSTEM
* BREAK THINGS **EXACTLY** A LITTLE BIT ON FAILURE

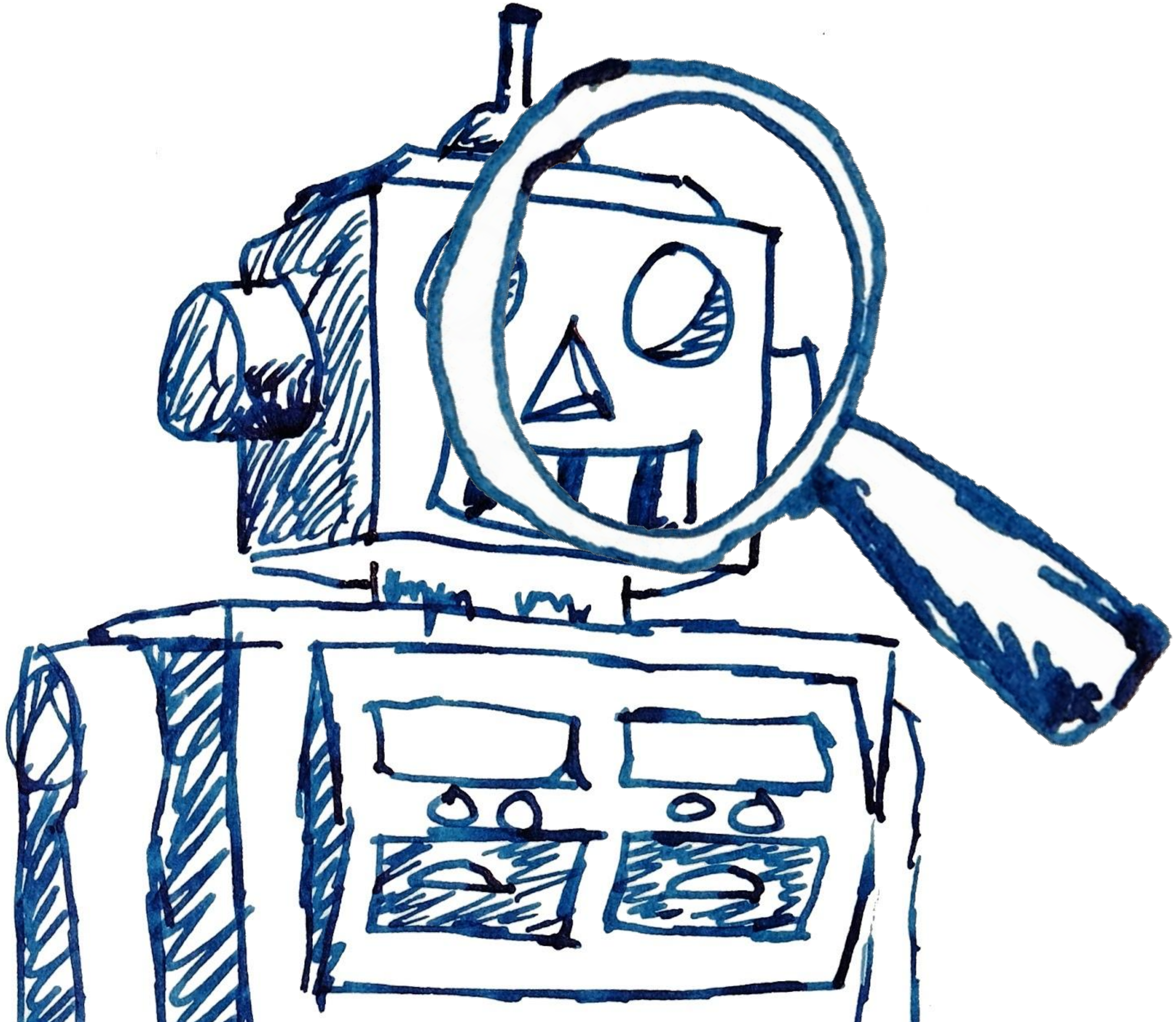


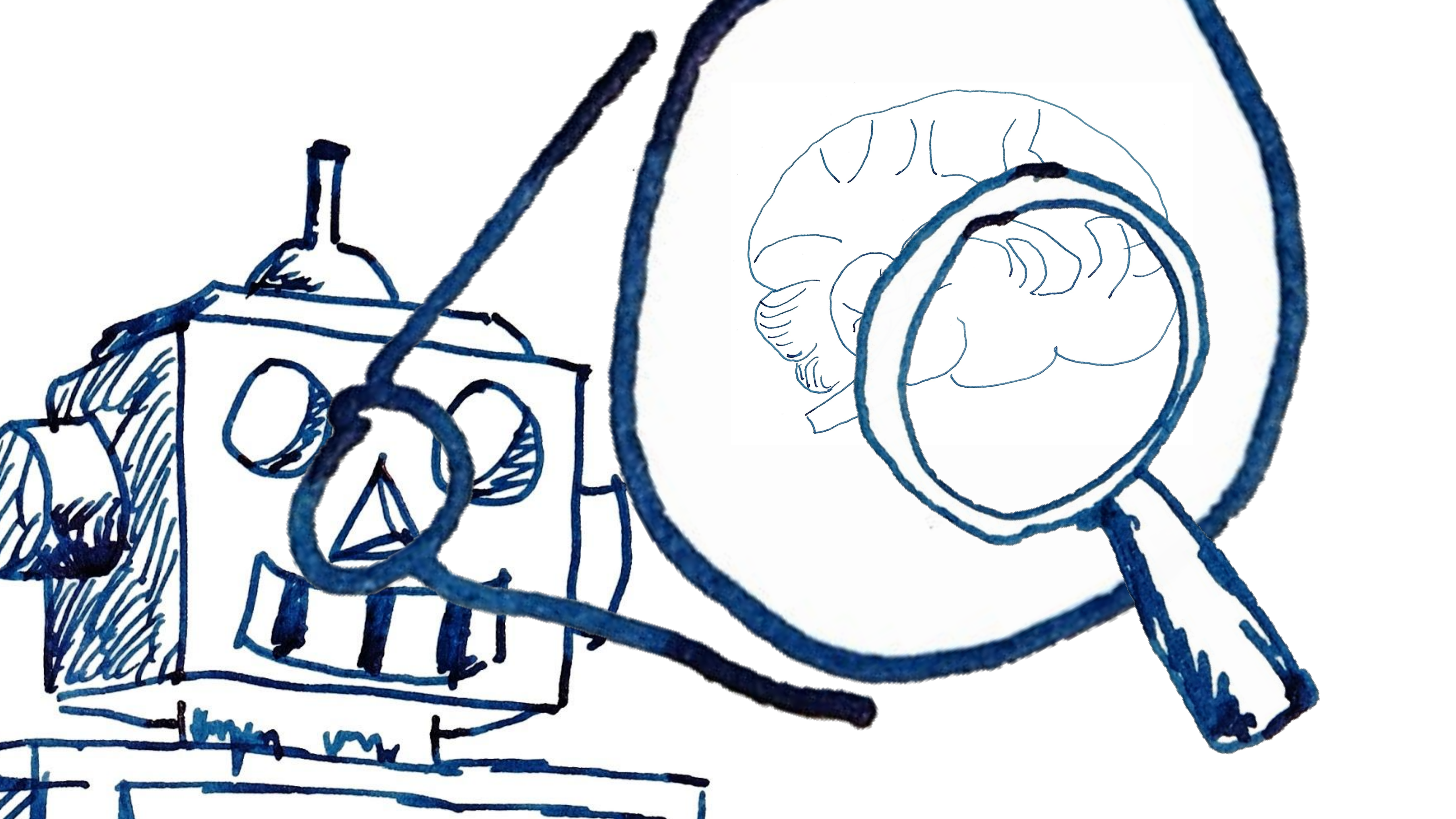

HI
FAN
LO
COOL
OFF
HEAT
50
60
70
80
90
90
80
70
60
50
THERMOSTAT

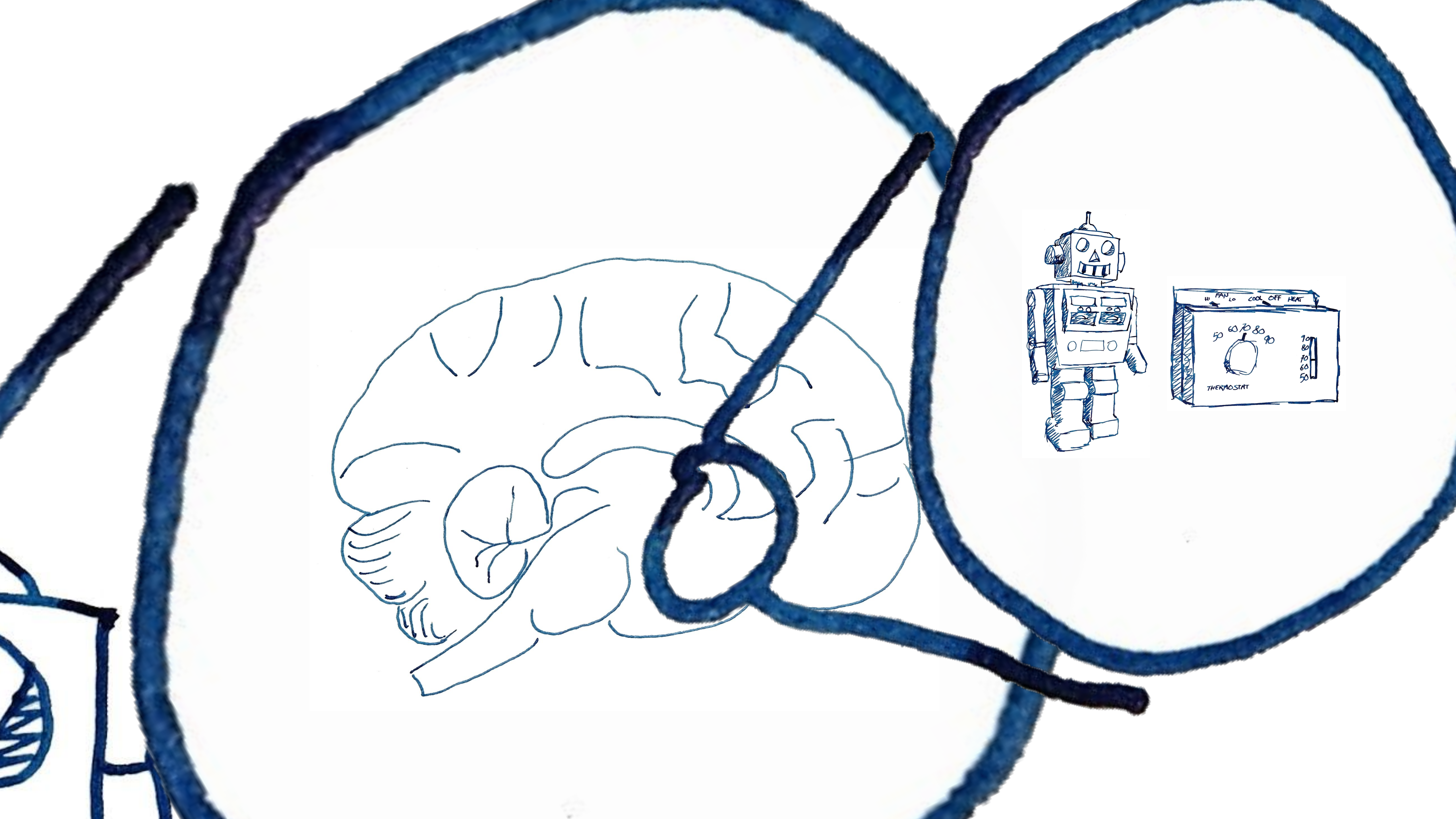



HI FAN LO
COOL OFF HEAT
50 60 70 80 90
90 80 70 60 50
THERMOSTAT

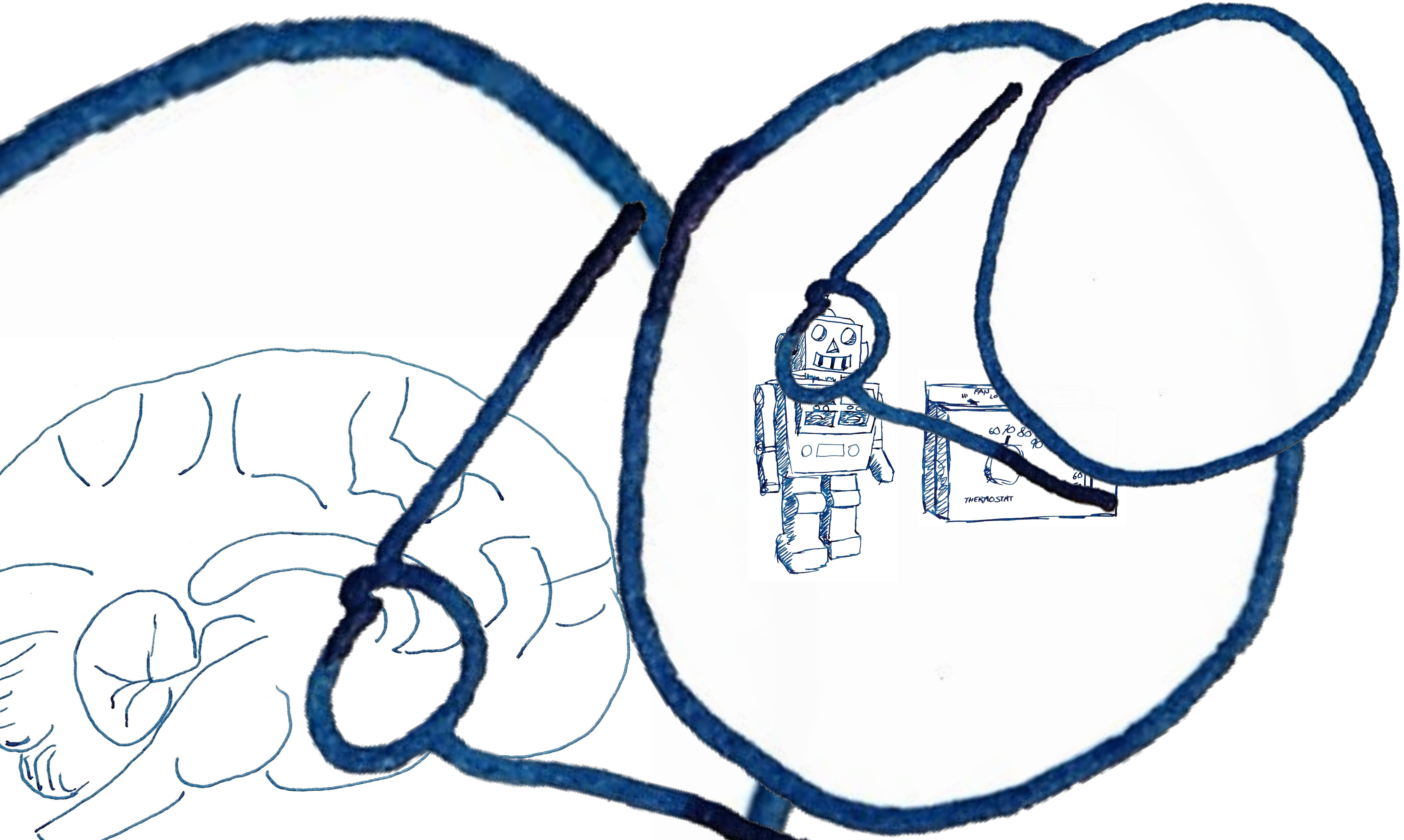







HI FAN LO
COOL OFF HEAT
50 60 70 80 90
90 80 70 60 50
THERMOSTAT

HI FAN LO COOL OFF HEAT
50 60 70 80 90
90 80 70 60 50
RMOSTAT


FAN
HI
LO
60 70 80 90
60
THERMOSTAT

AN IDEAL KRM CONTROLLER SHOULD:

* ONLY DO **ONE** THING
* HAVE AN **INPUT** SOURCE
* HAVE A PLACE TO WRITE **STATUS**
* HAVE AN **OUTPUT** LOCATION
* **ANTICIPATE ITS OWN EFFECTS** ON THE REST OF THE SYSTEM
* BREAK THINGS **EXACTLY** A LITTLE BIT ON FAILURE

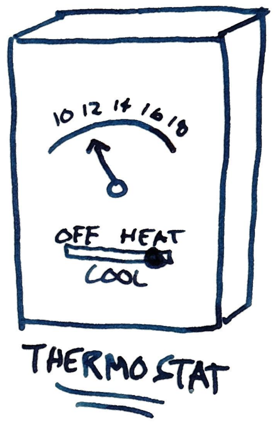
CONTROL ~~THEORY~~
PRACTICE!

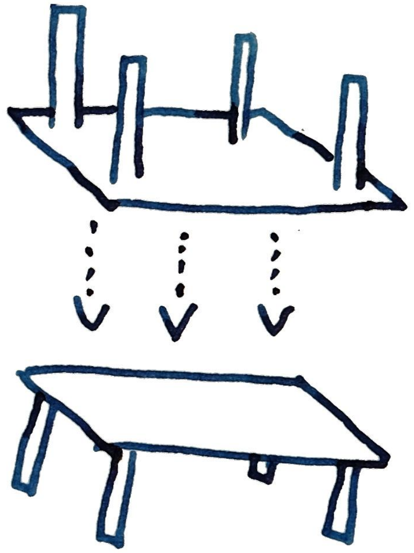
What kinds of controllers are there?

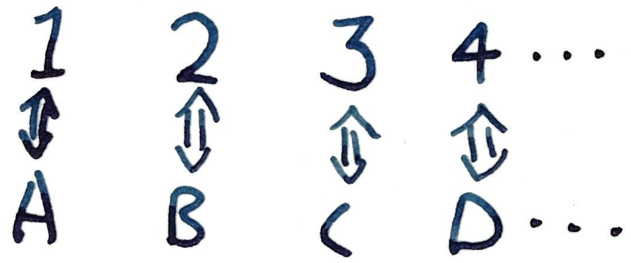
- The “classic” controllers
- Bi- or injection enforcers
- Standing query / table join

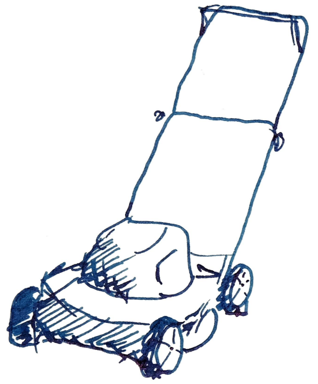
CONTROLLER CATEGORIES

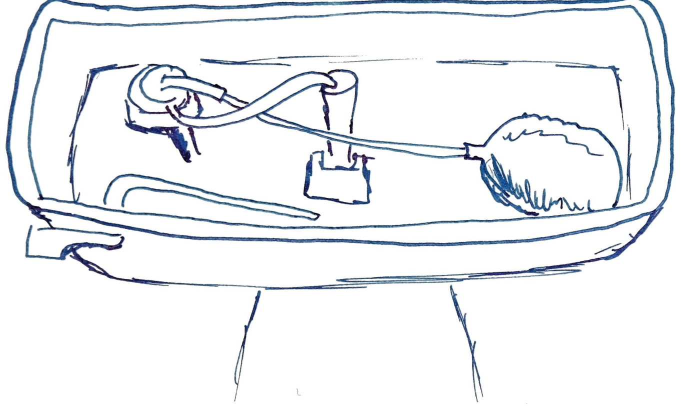
* THE "CLASSIC" CONTROLLERS
* STANDING QUERY / "TABLE JOIN"
* IN- OR BIJECTION ENFORCERS

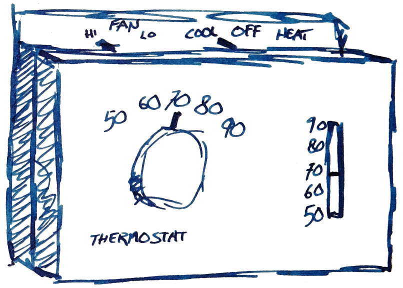


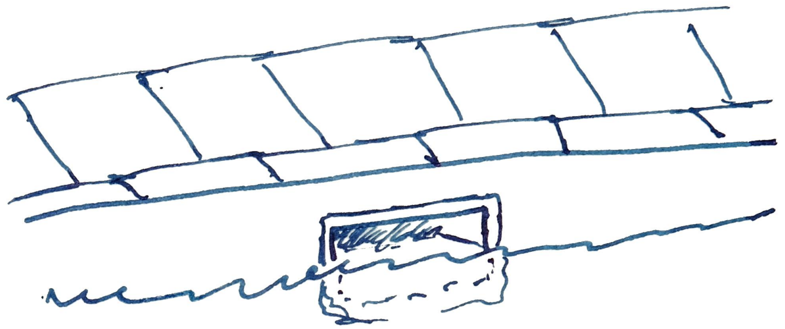


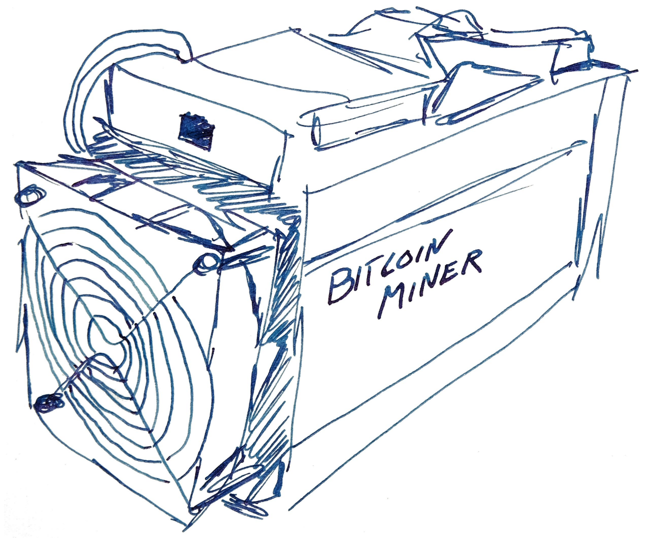


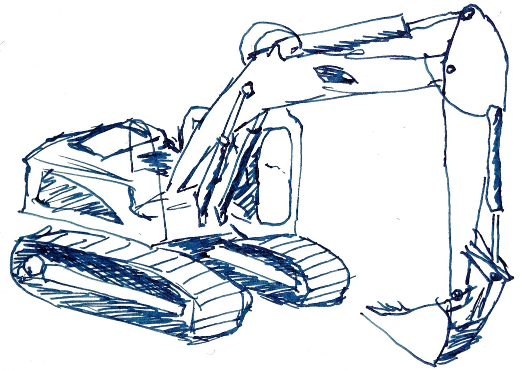


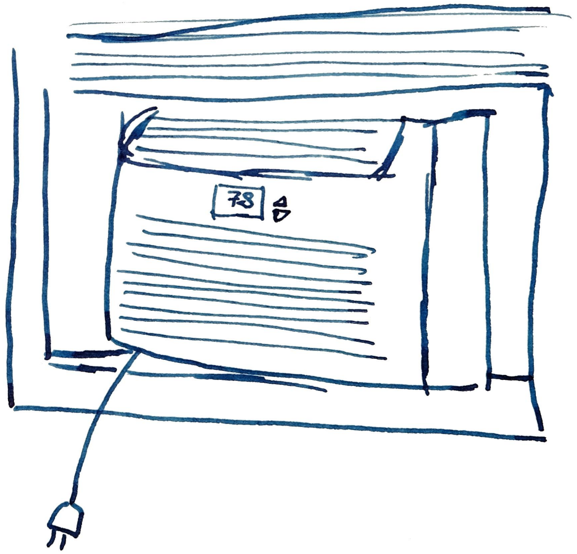






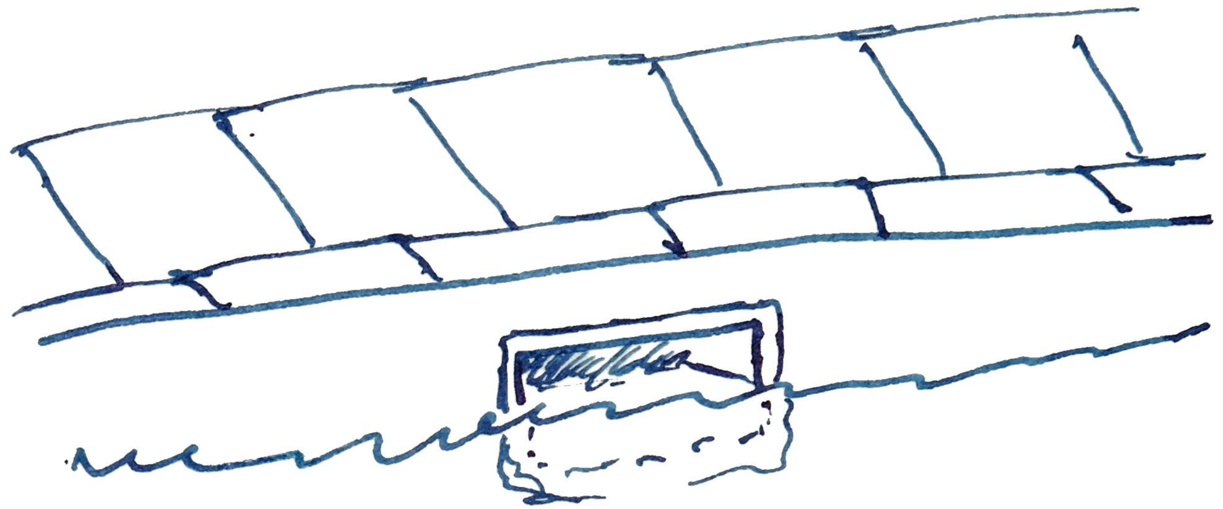


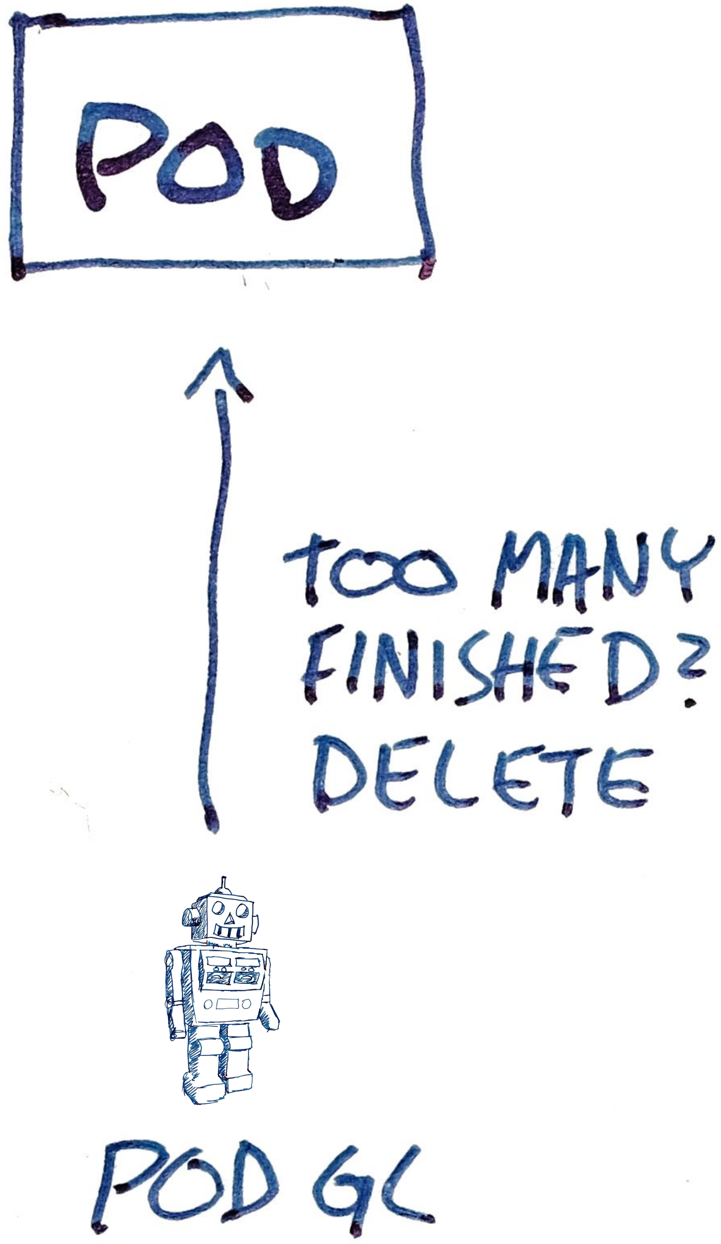




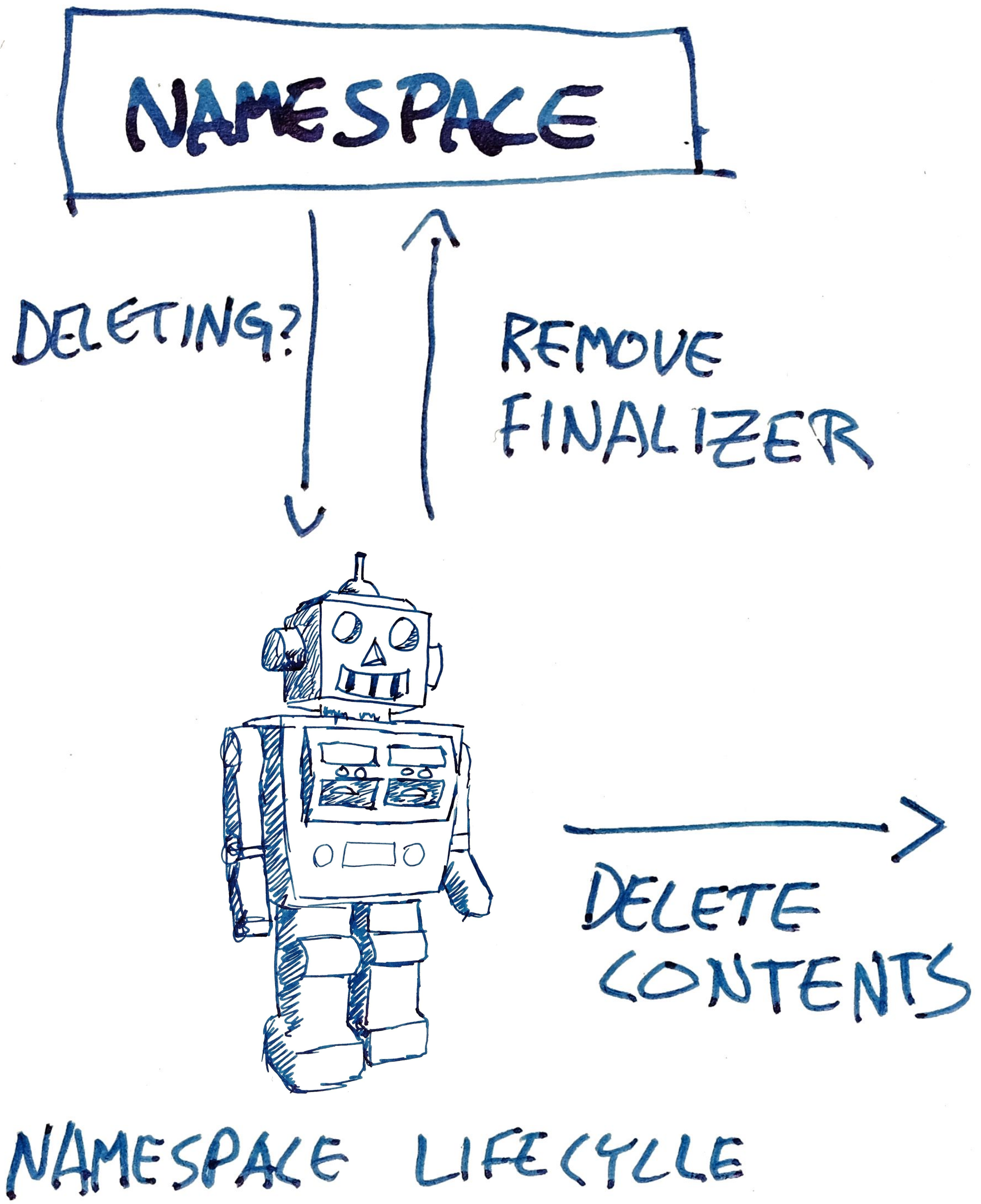
CLASSIC CONTROLLER

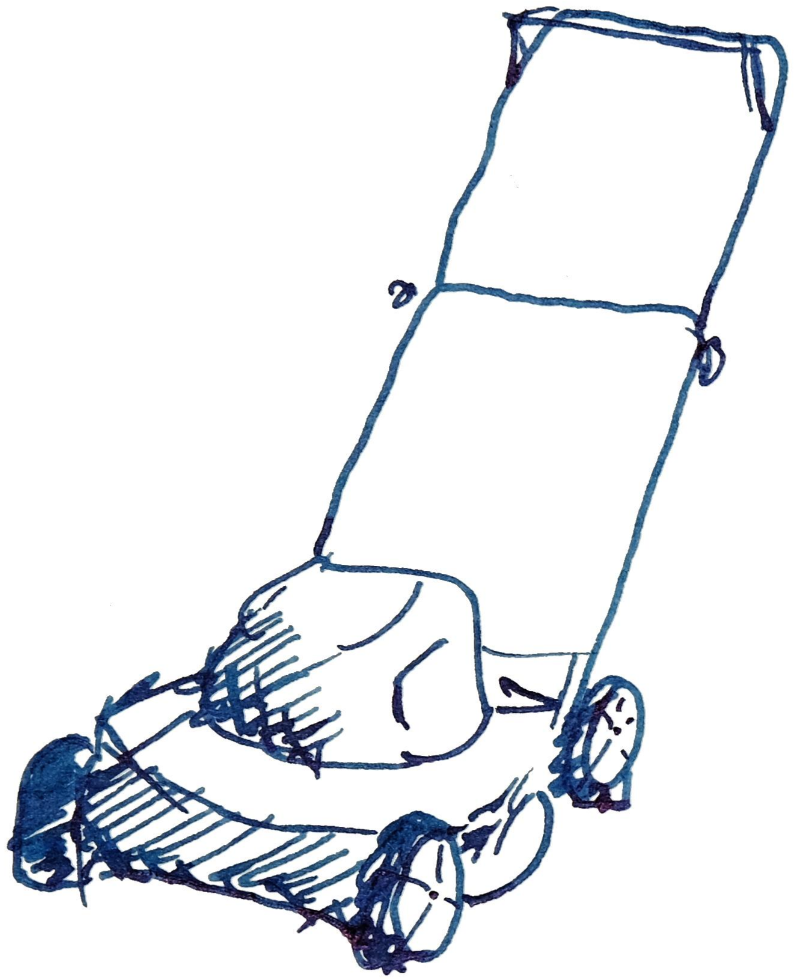
CLASSIC CONTROLLER



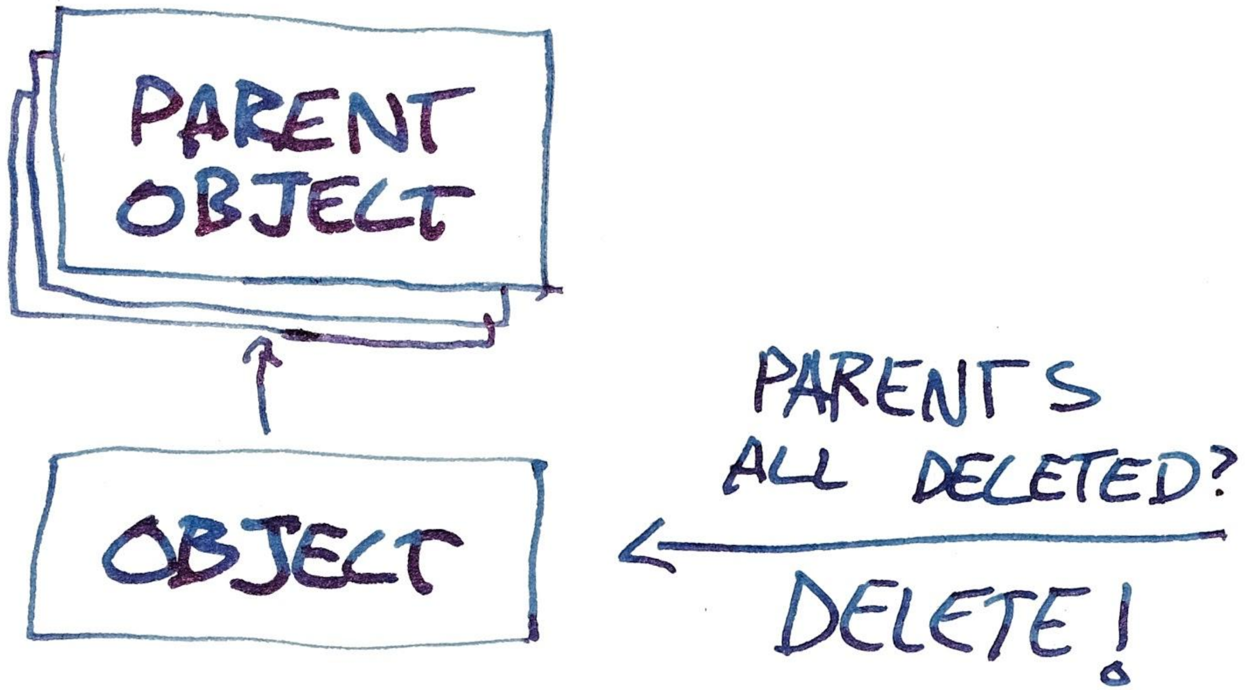


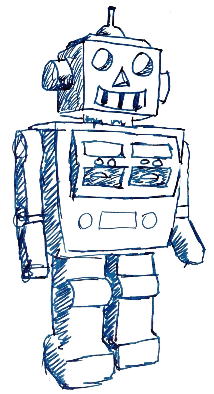
CLASSIC CONTROLLER





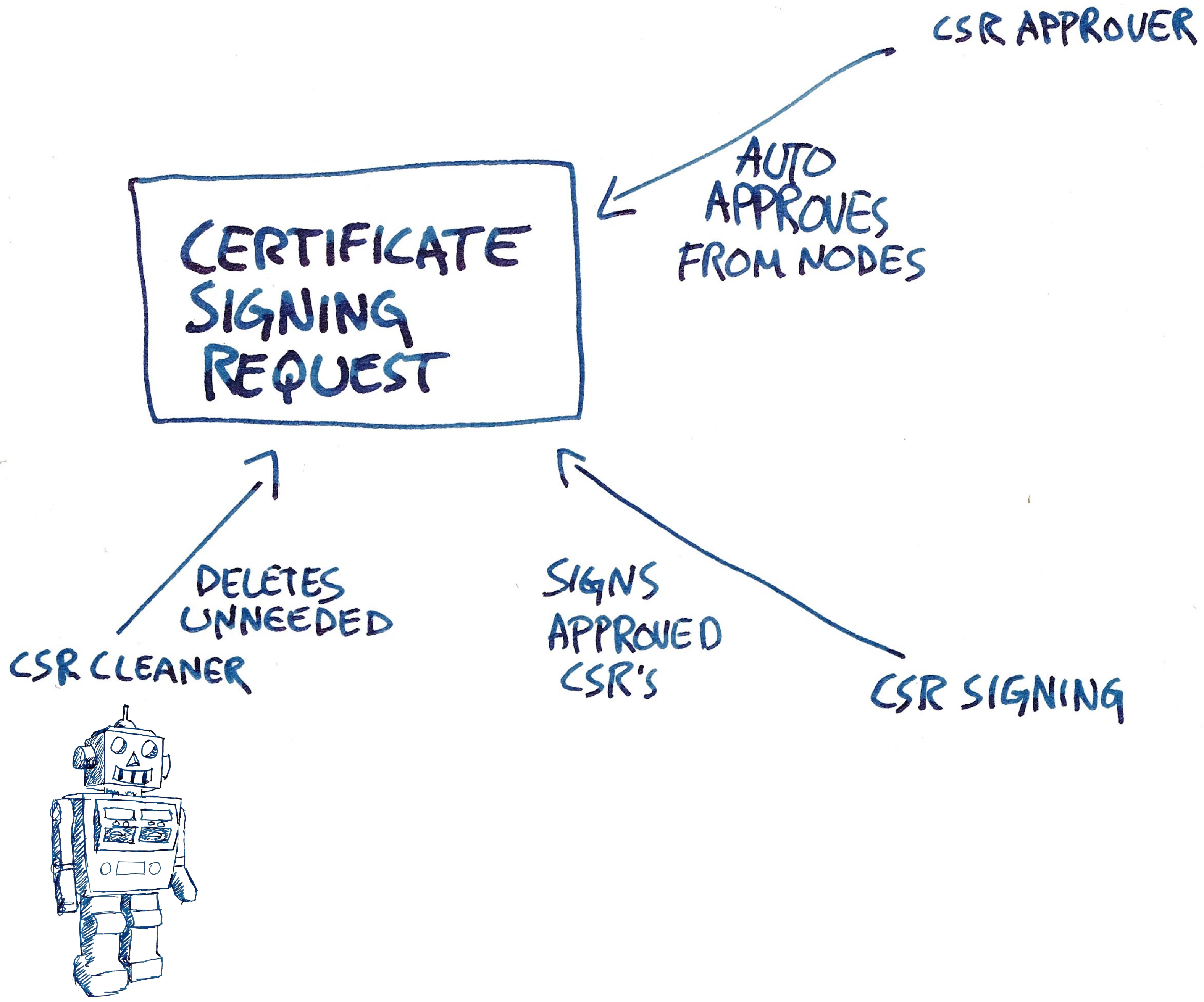
CLASSIC CONTROLLER



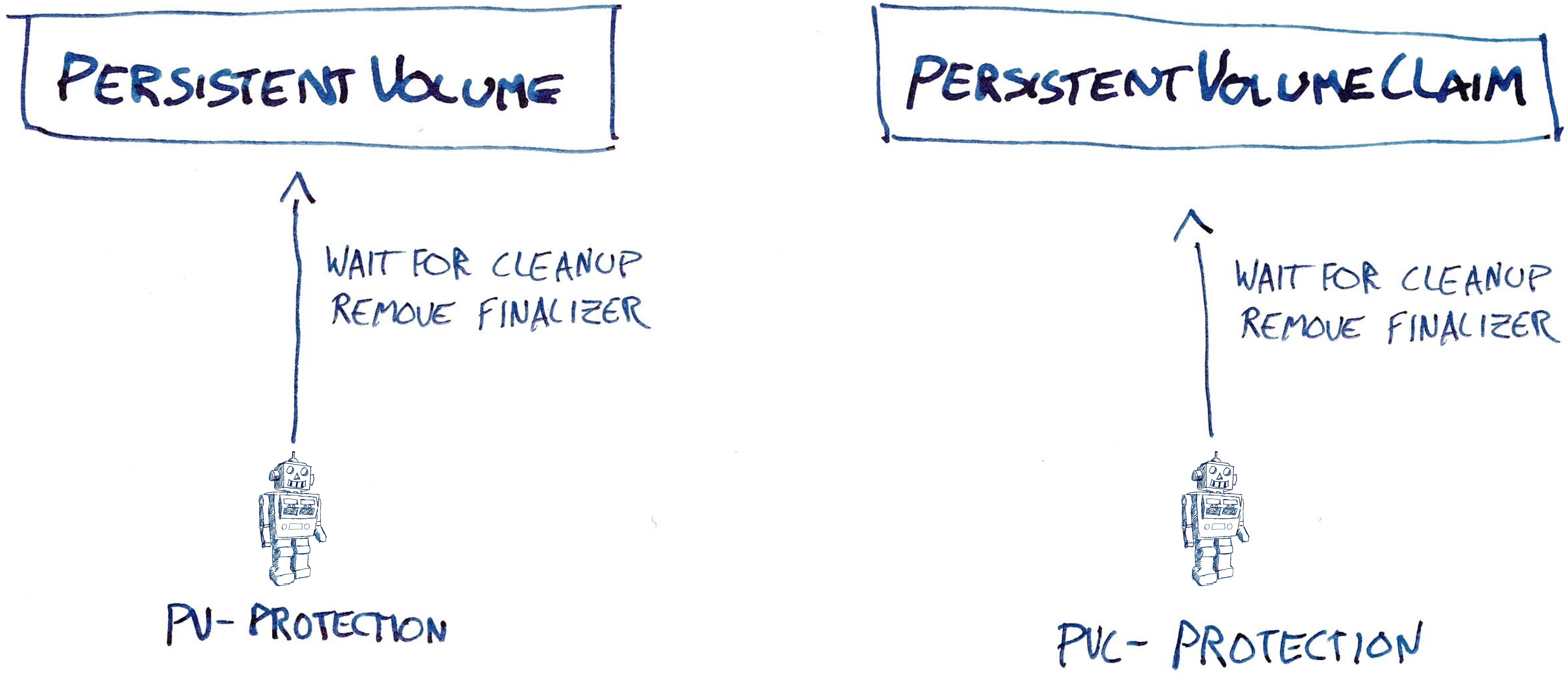


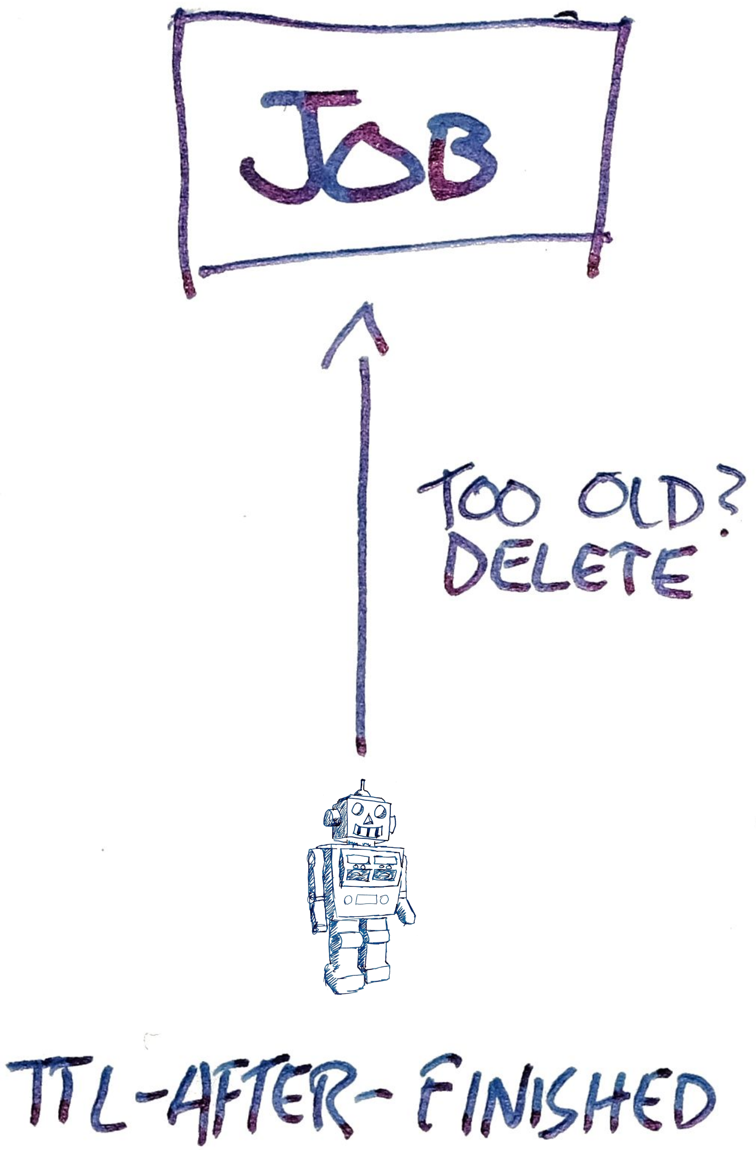
GARBAGE COLLECTOR

CLASSIC CONTROLLER

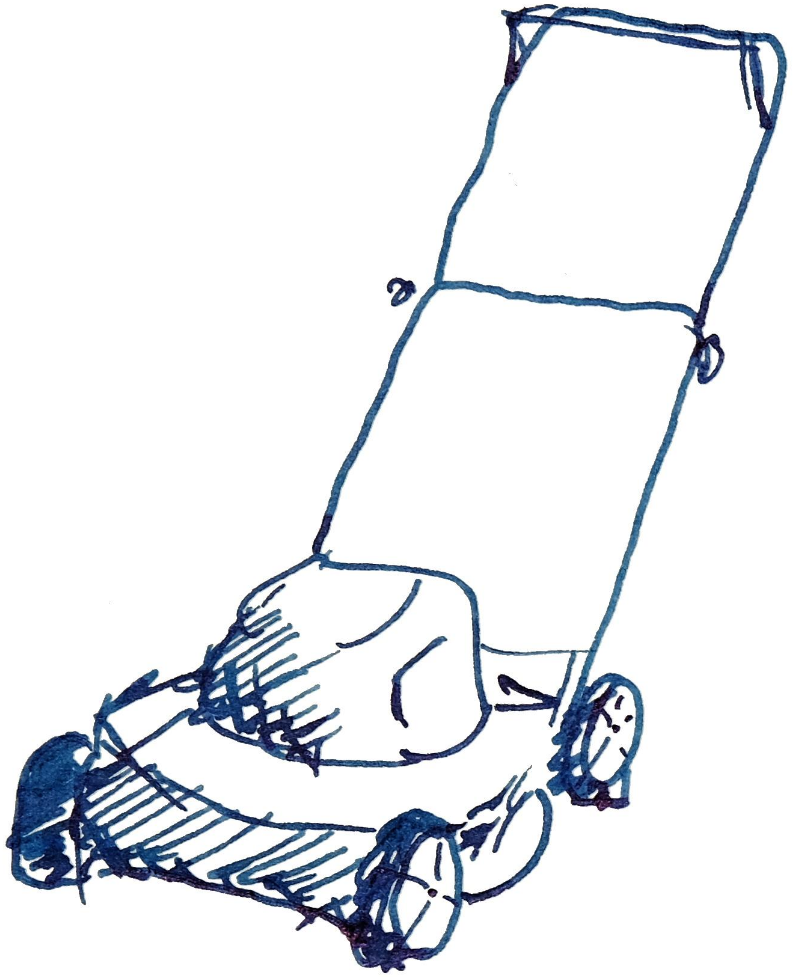


CLASSIC CONTROLLER

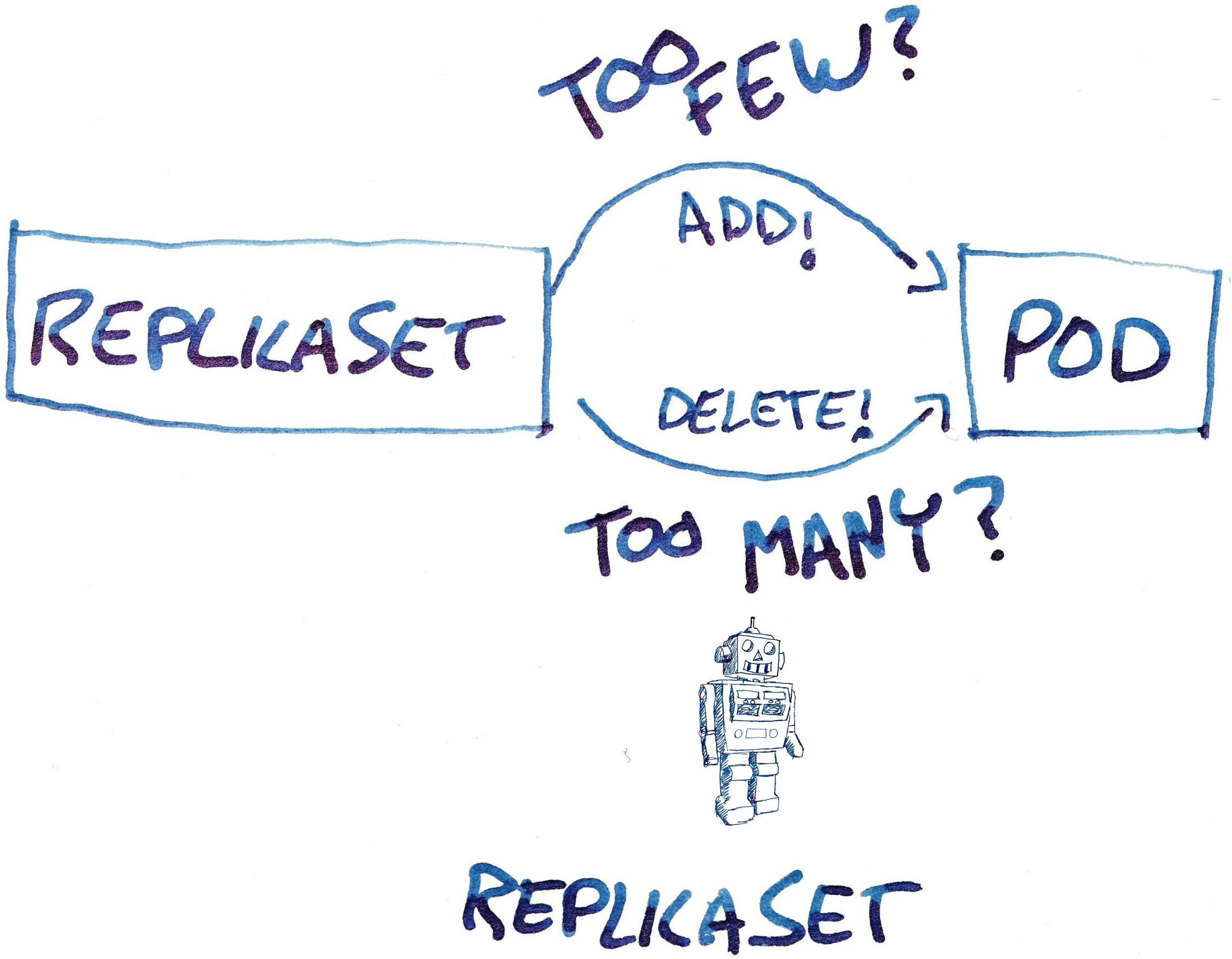


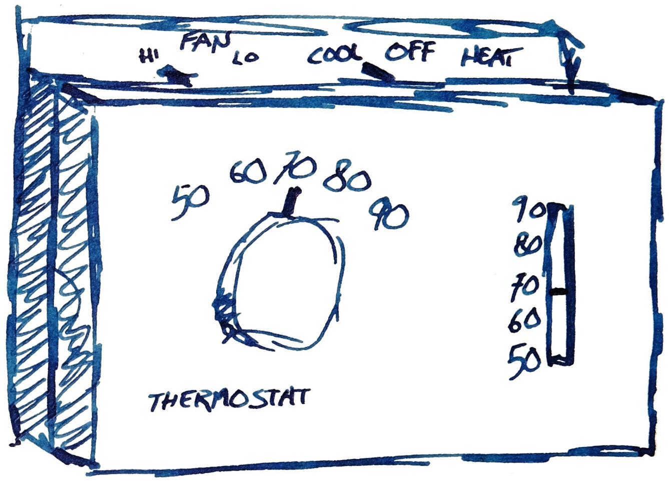


CLASSIC CONTROLLER

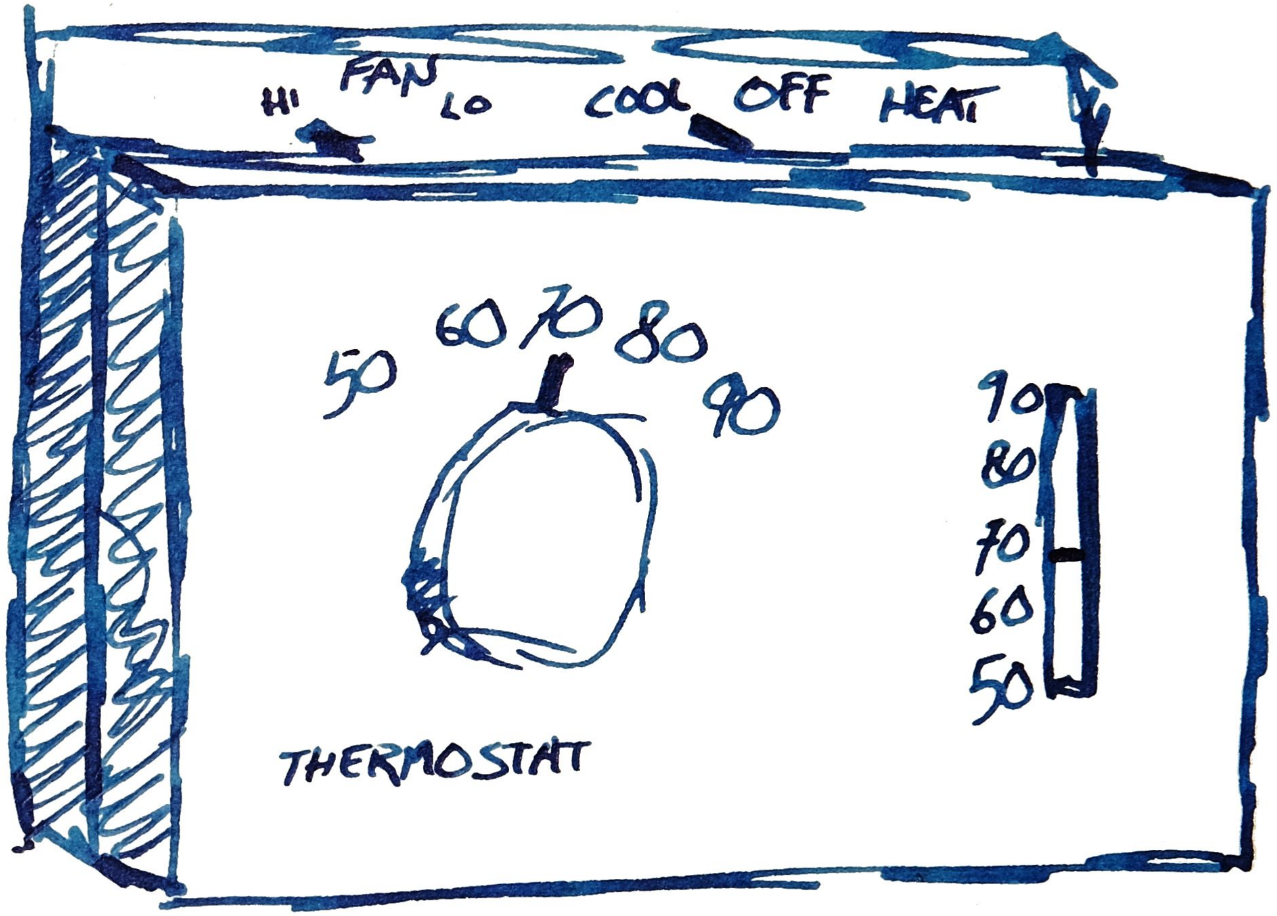


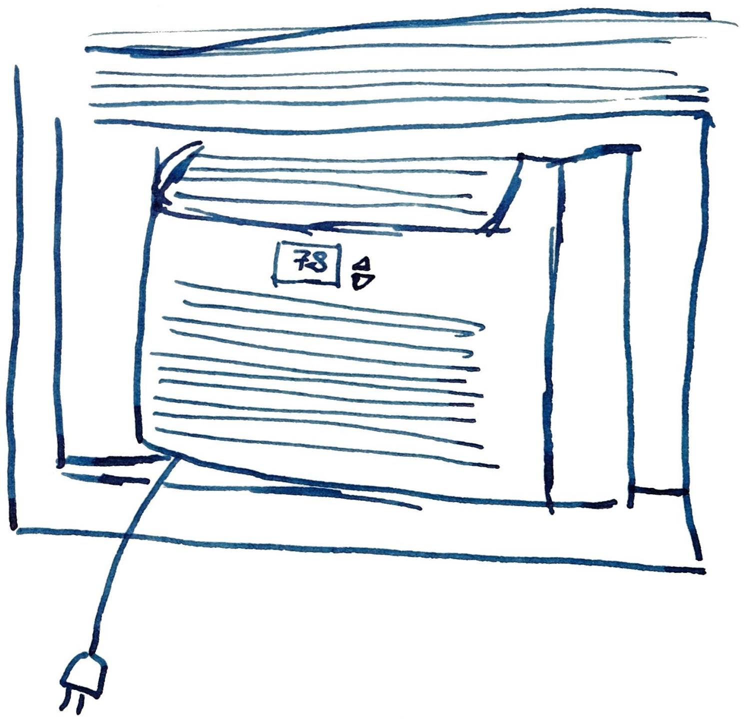
CLASSIC CONTROLLER

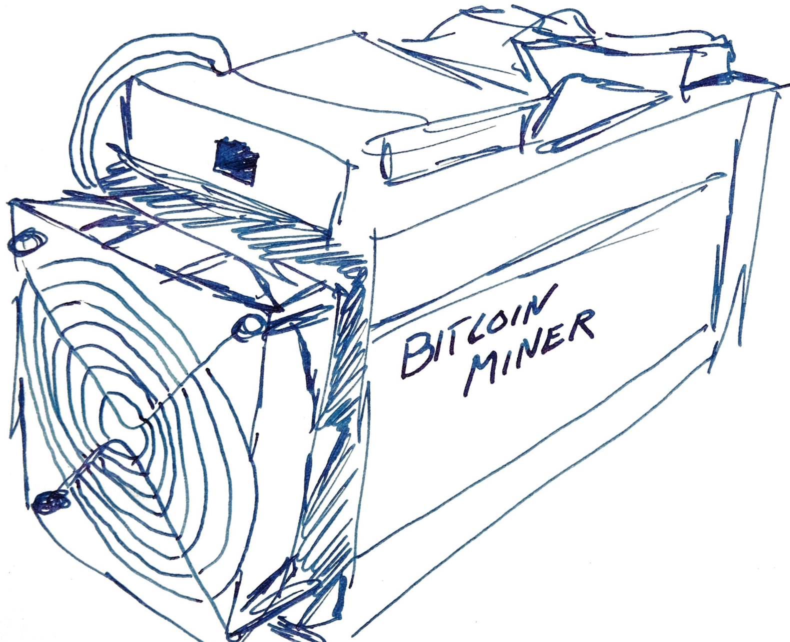




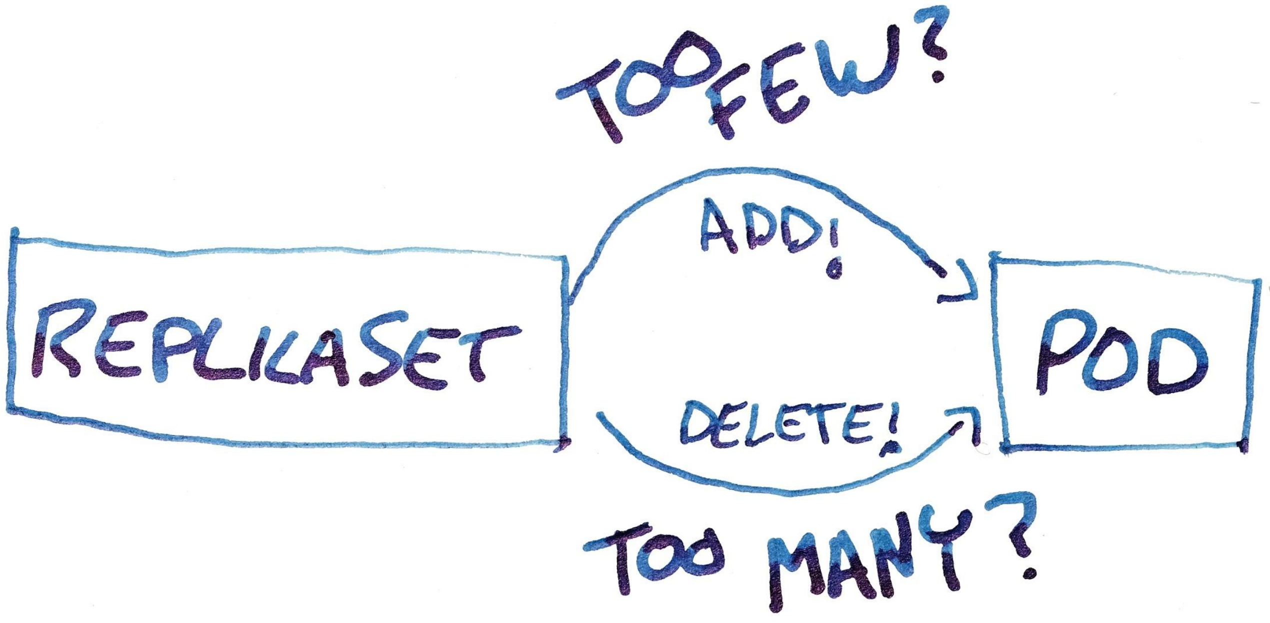
CLASSIC CONTROLLER

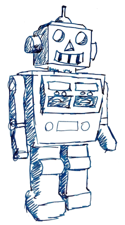




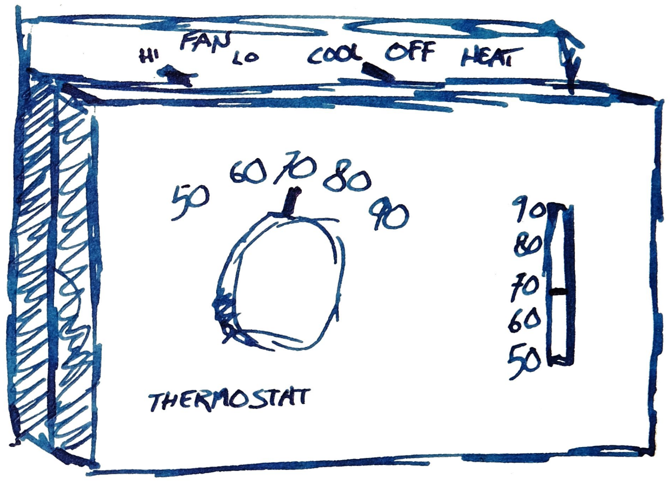


CLASSIC CONTROLLER

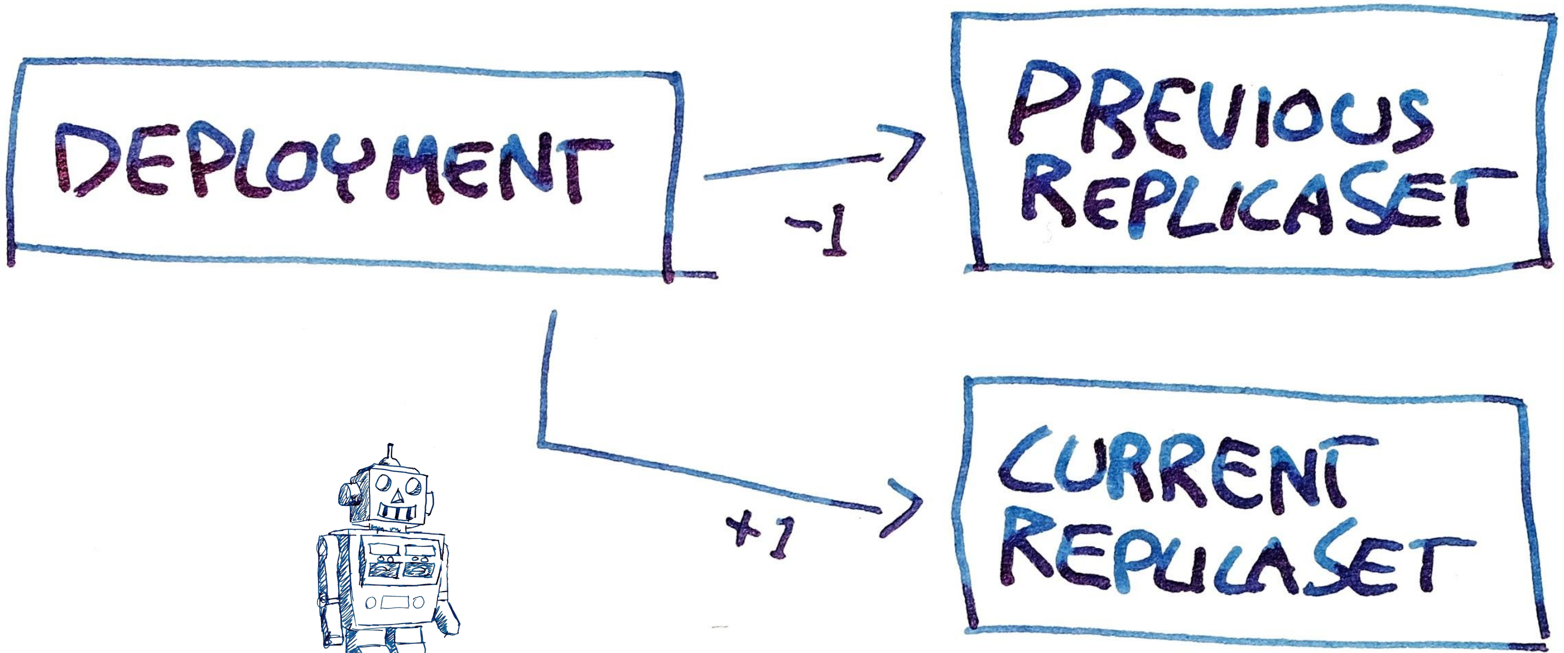


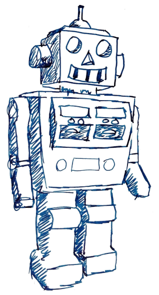


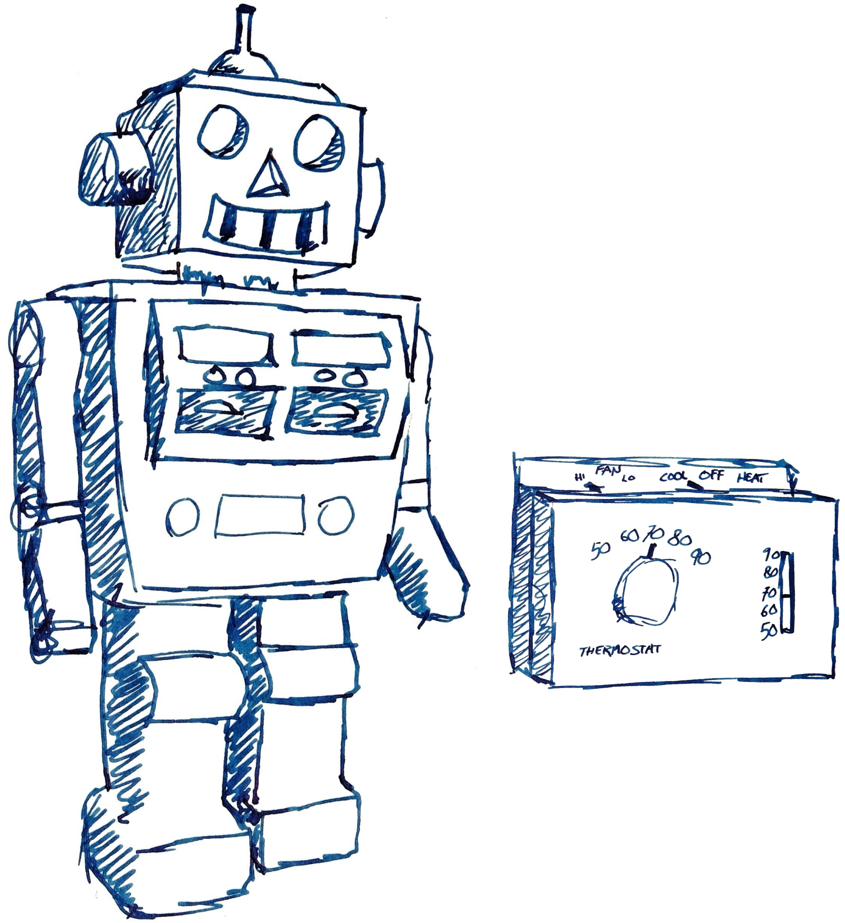
REPLICASET



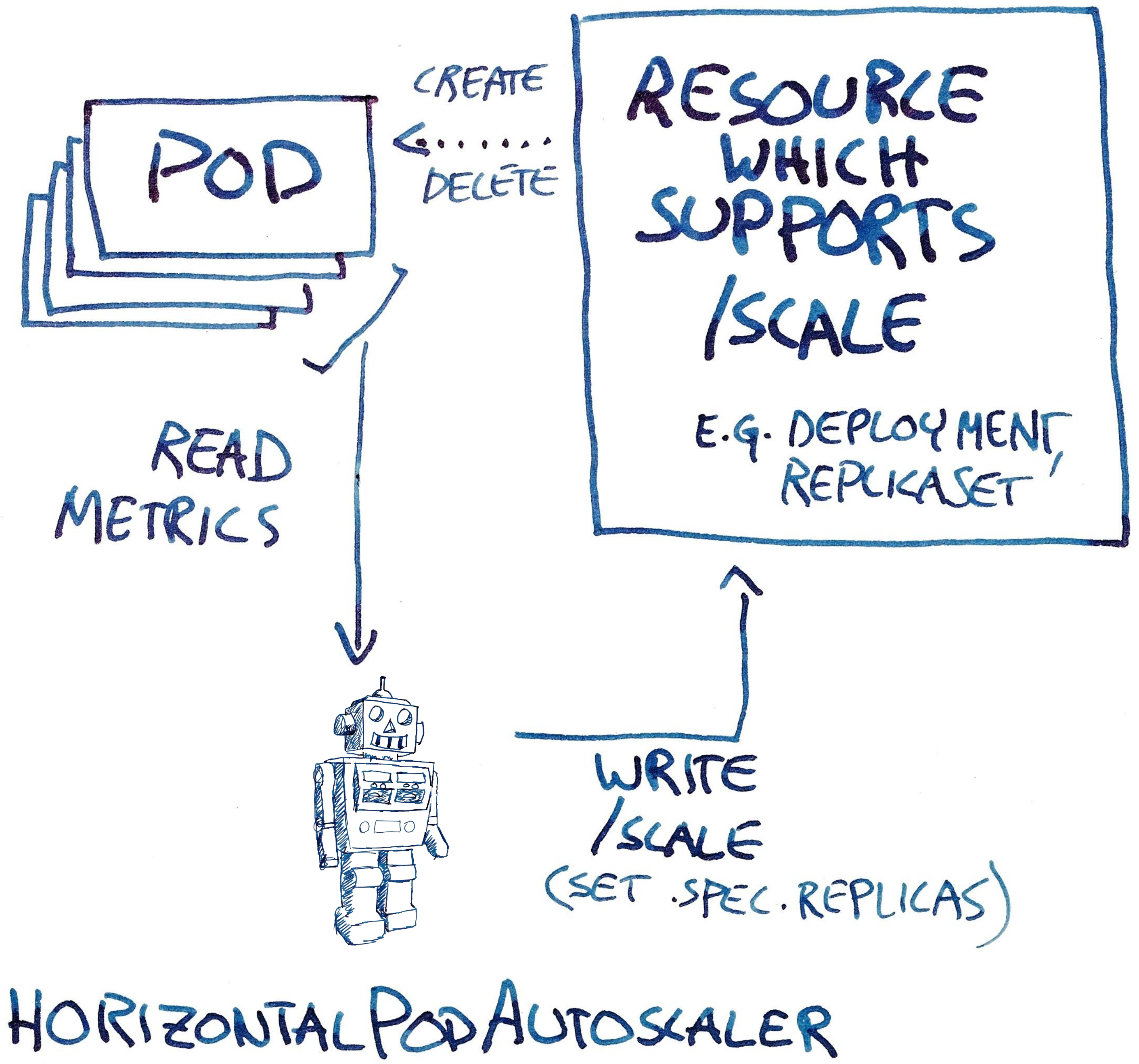
CLASSIC CONTROLLER

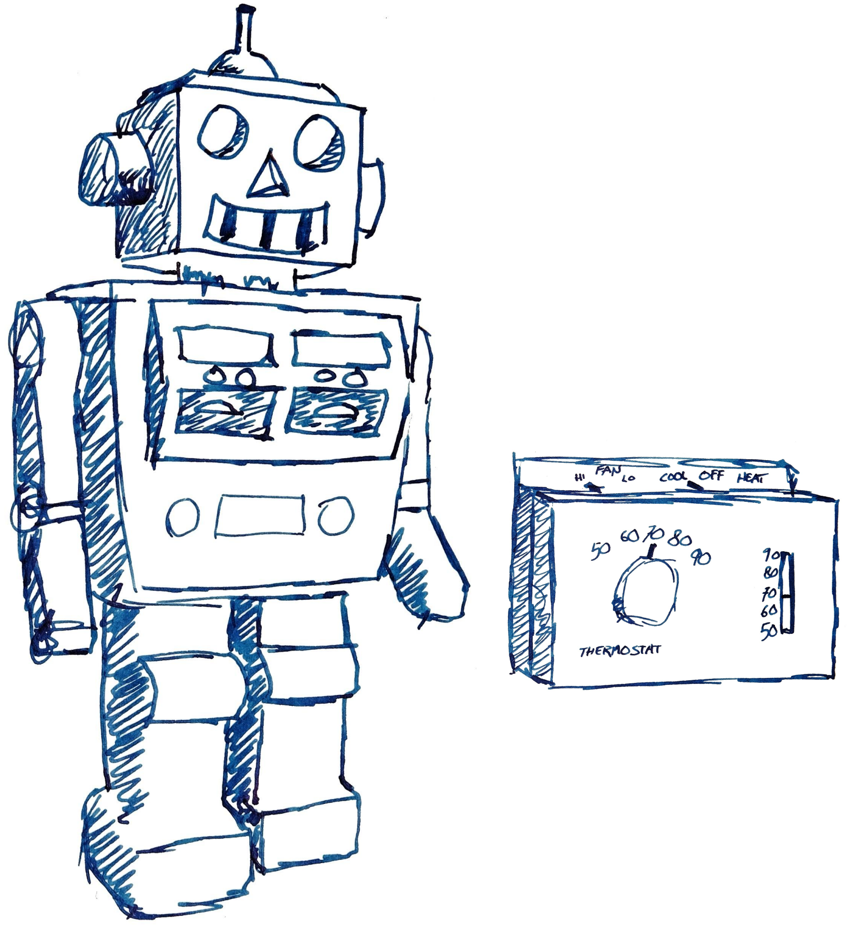



DEPLOYMENT

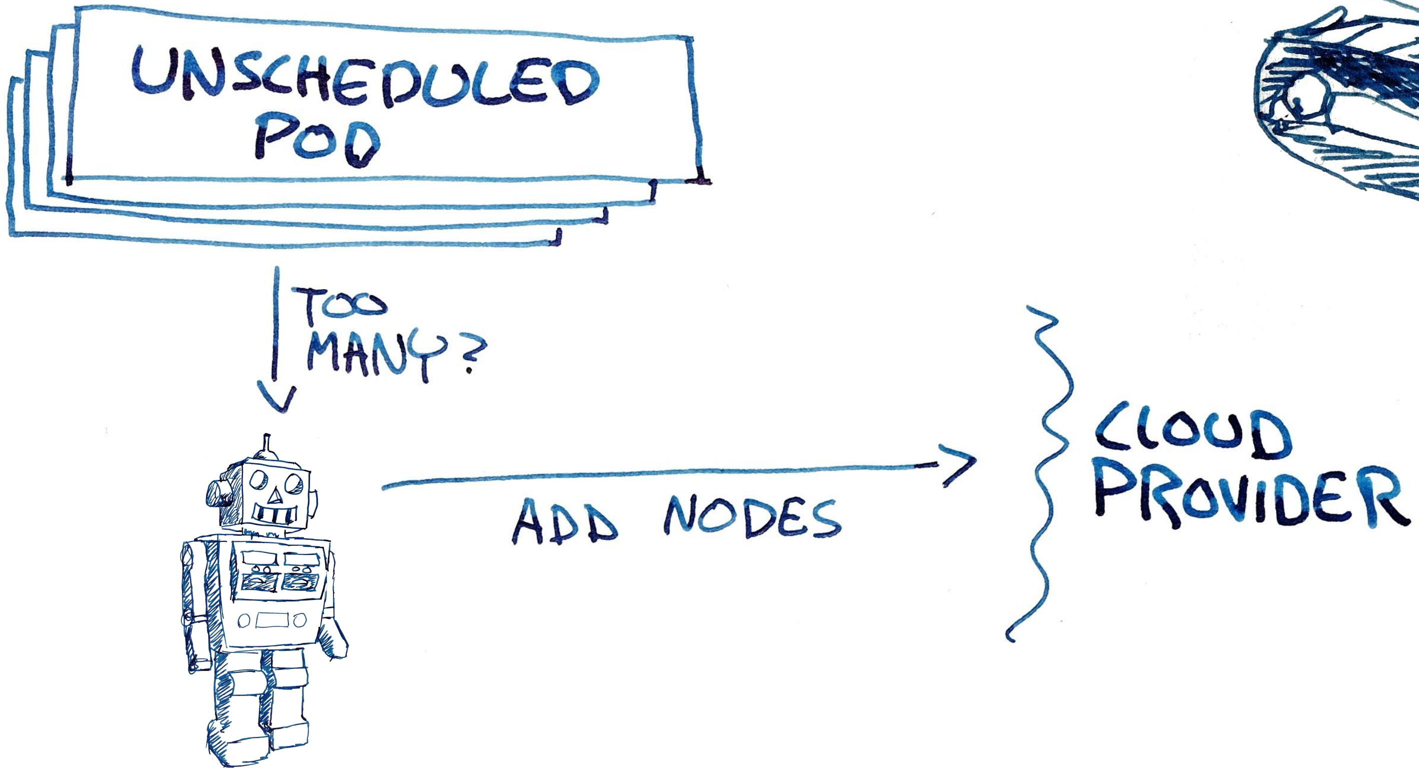


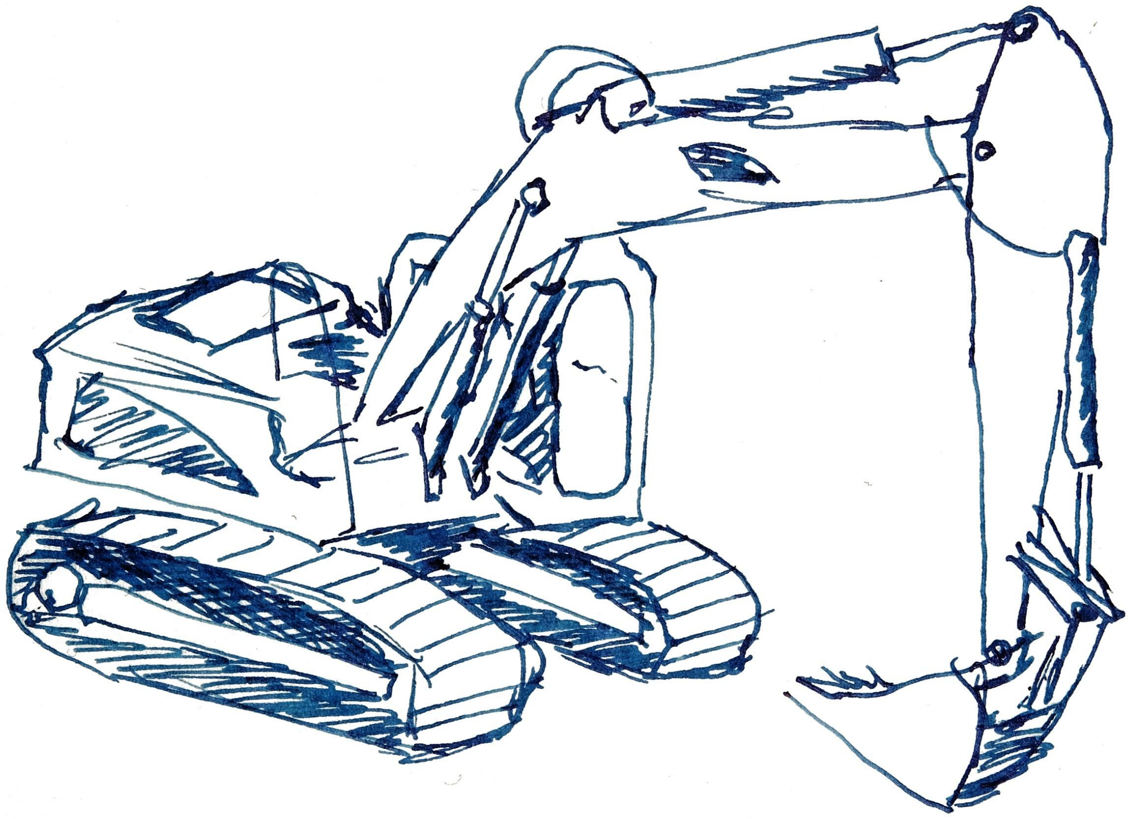
CLASSIC CONTROLLER


POD
CREATE
DELETE
RESOURCE WHICH SUPPORTS /SCALE
E.G. DEPLOYMENT, REPLICASET
READ METRICS
WRITE /SCALE (SET .SPEC.REPLICAS)
HORIZONTALPODAUTOSCALER

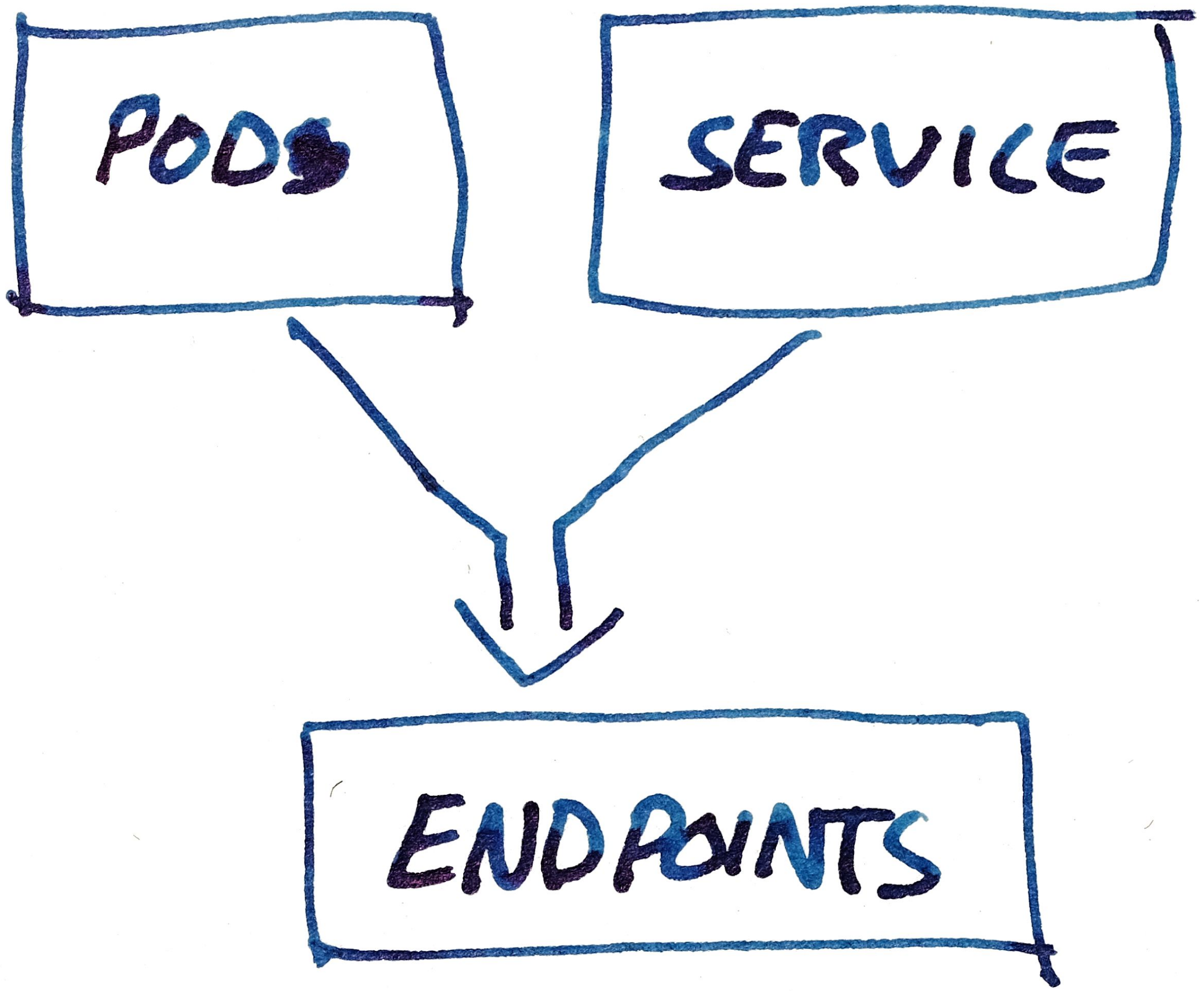


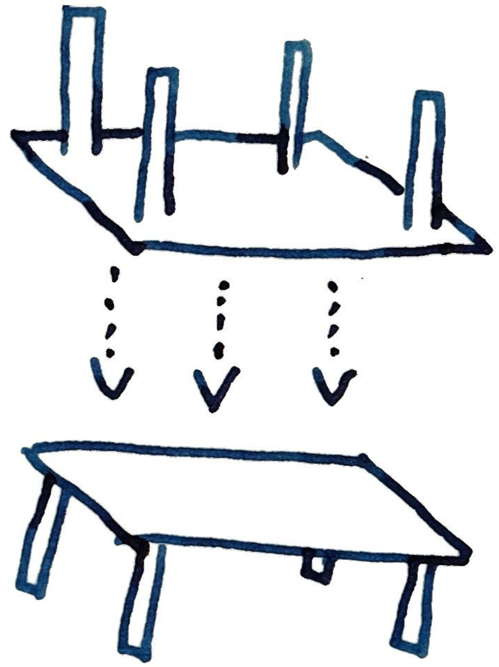
CLASSIC CONTROLLER



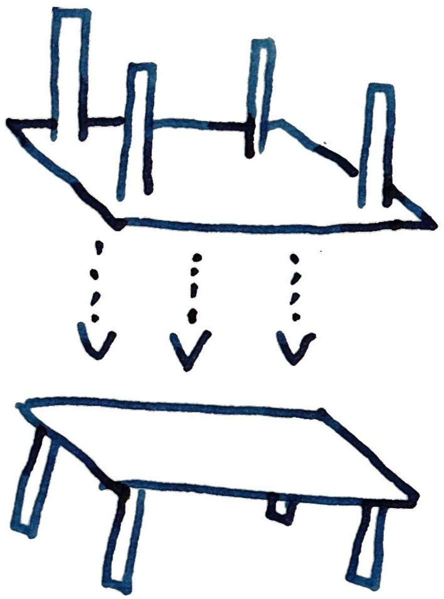


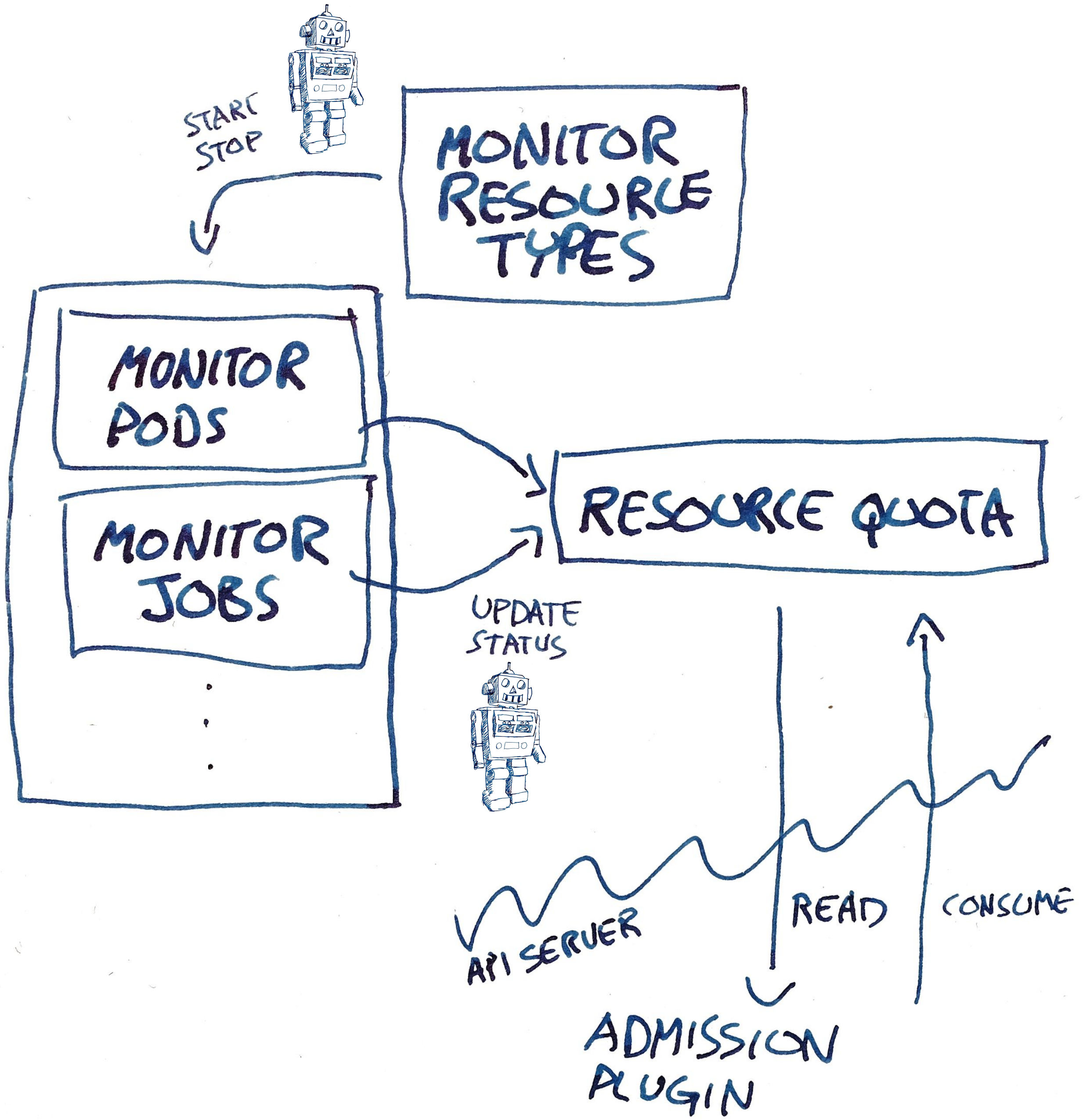
STANDING QUERY TABLE JOIN



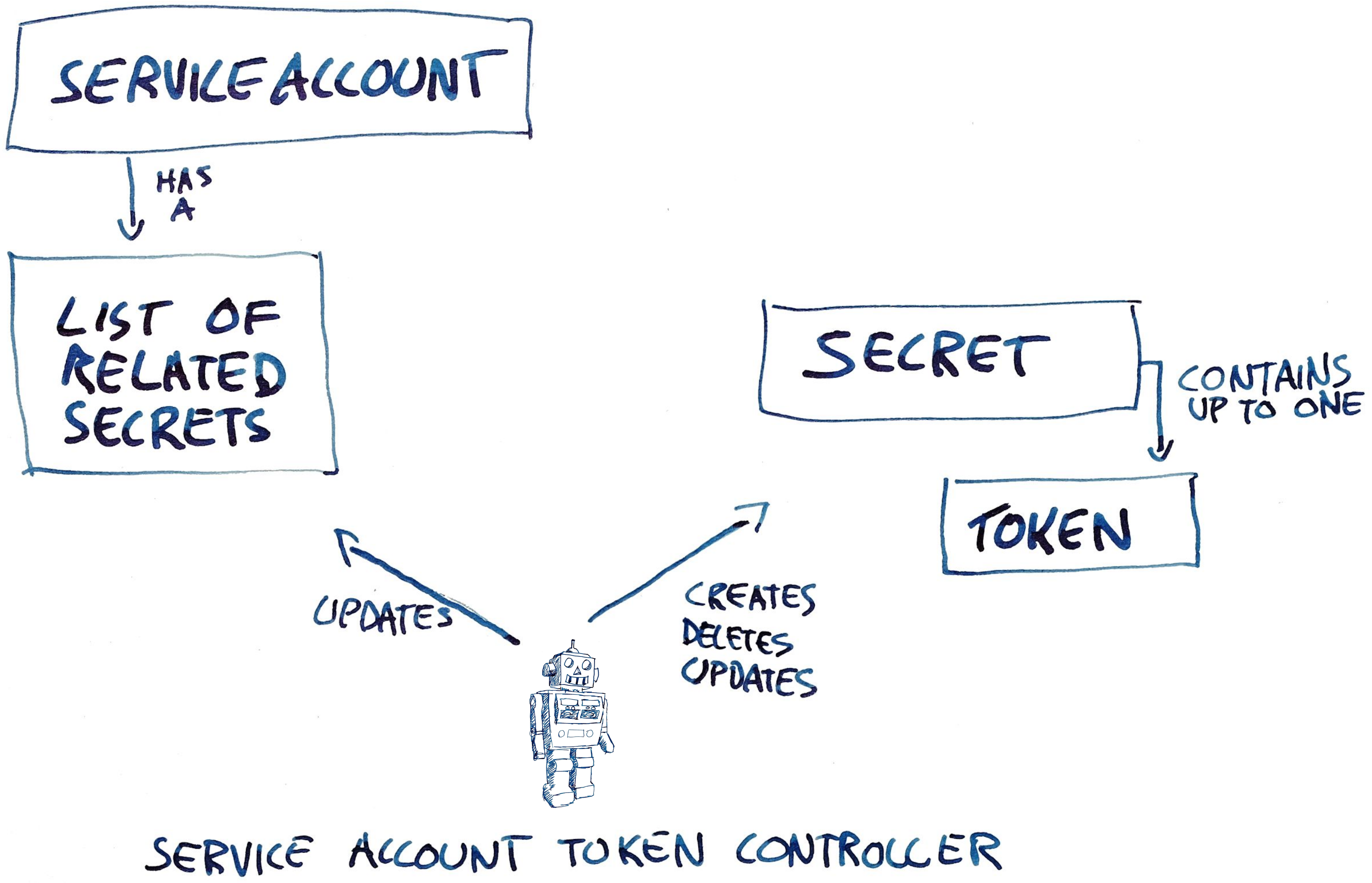


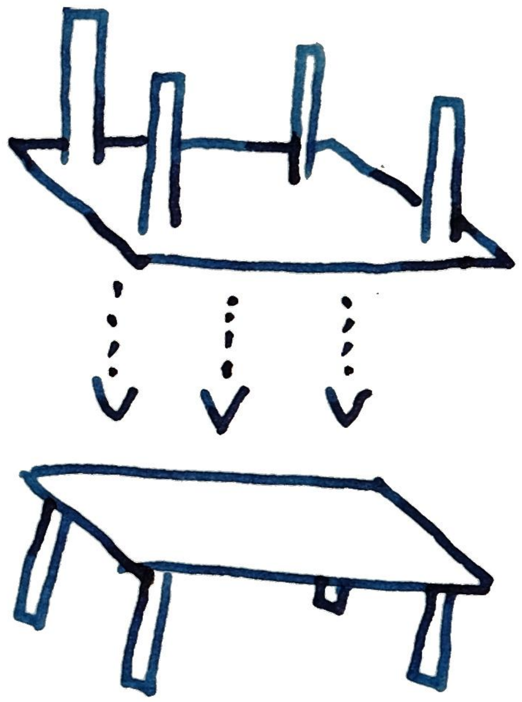
STANDING QUERY TABLE JOIN



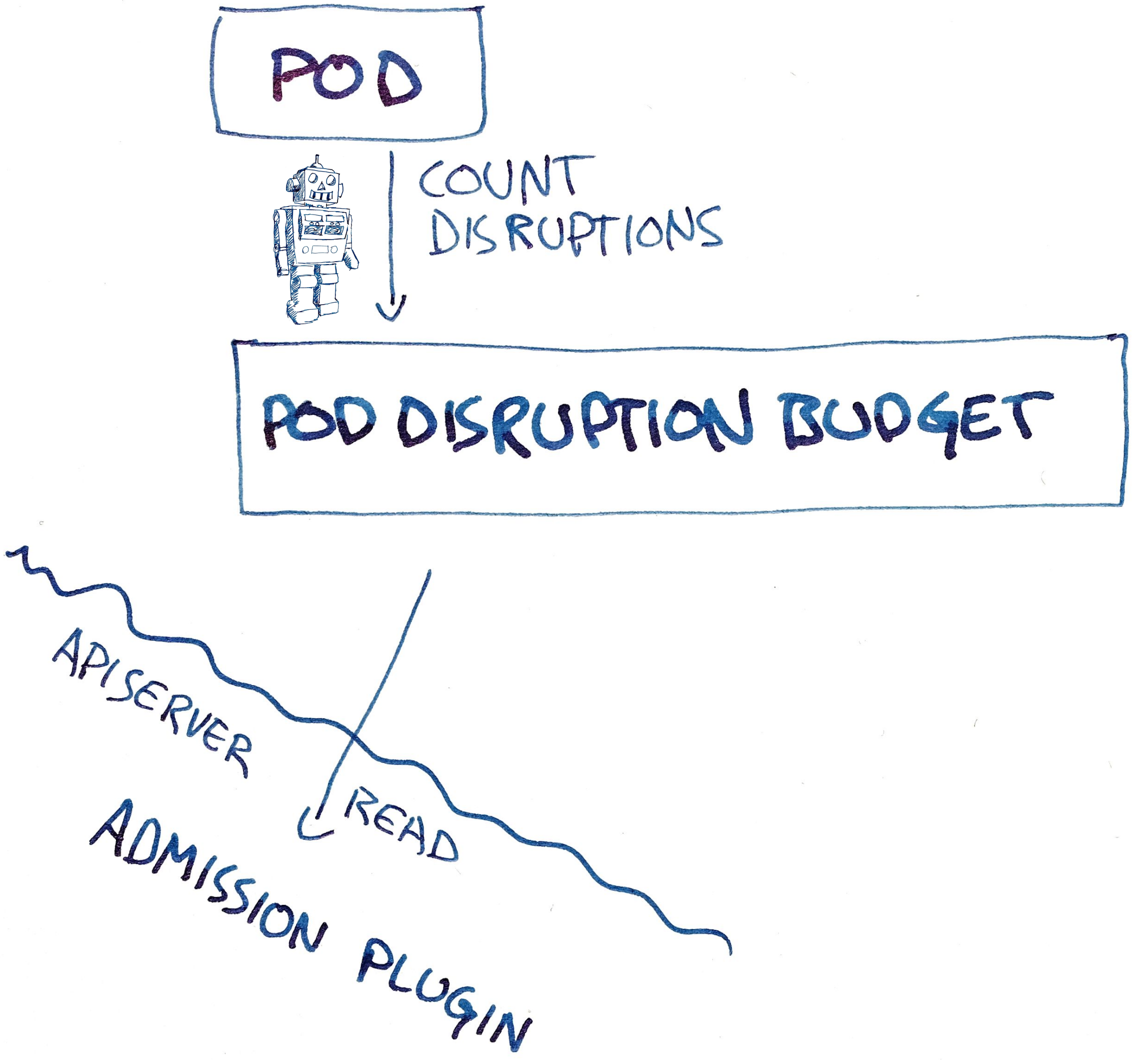


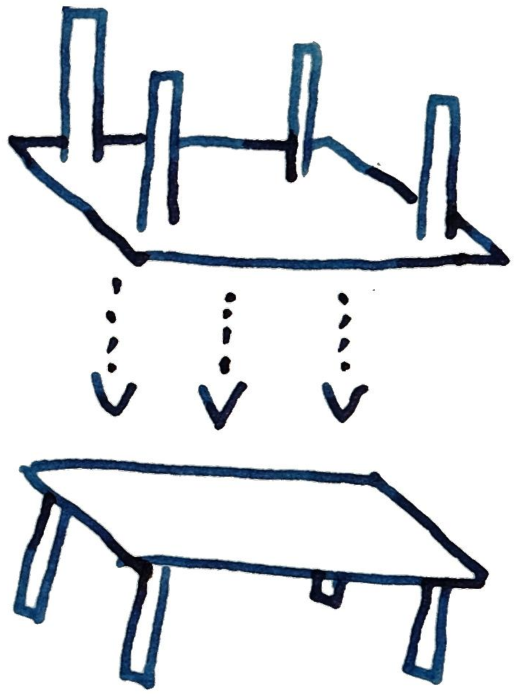
STANDING QUERY TABLE JOIN



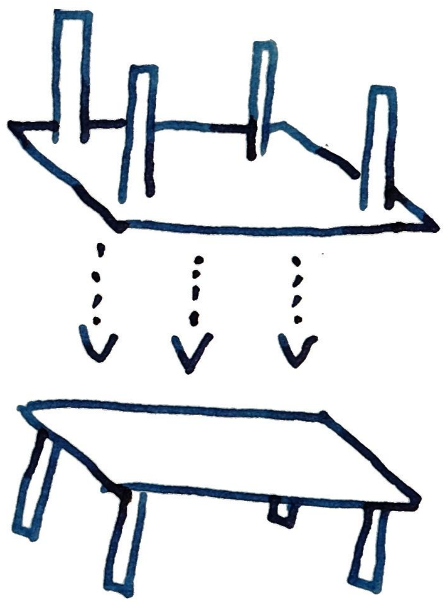


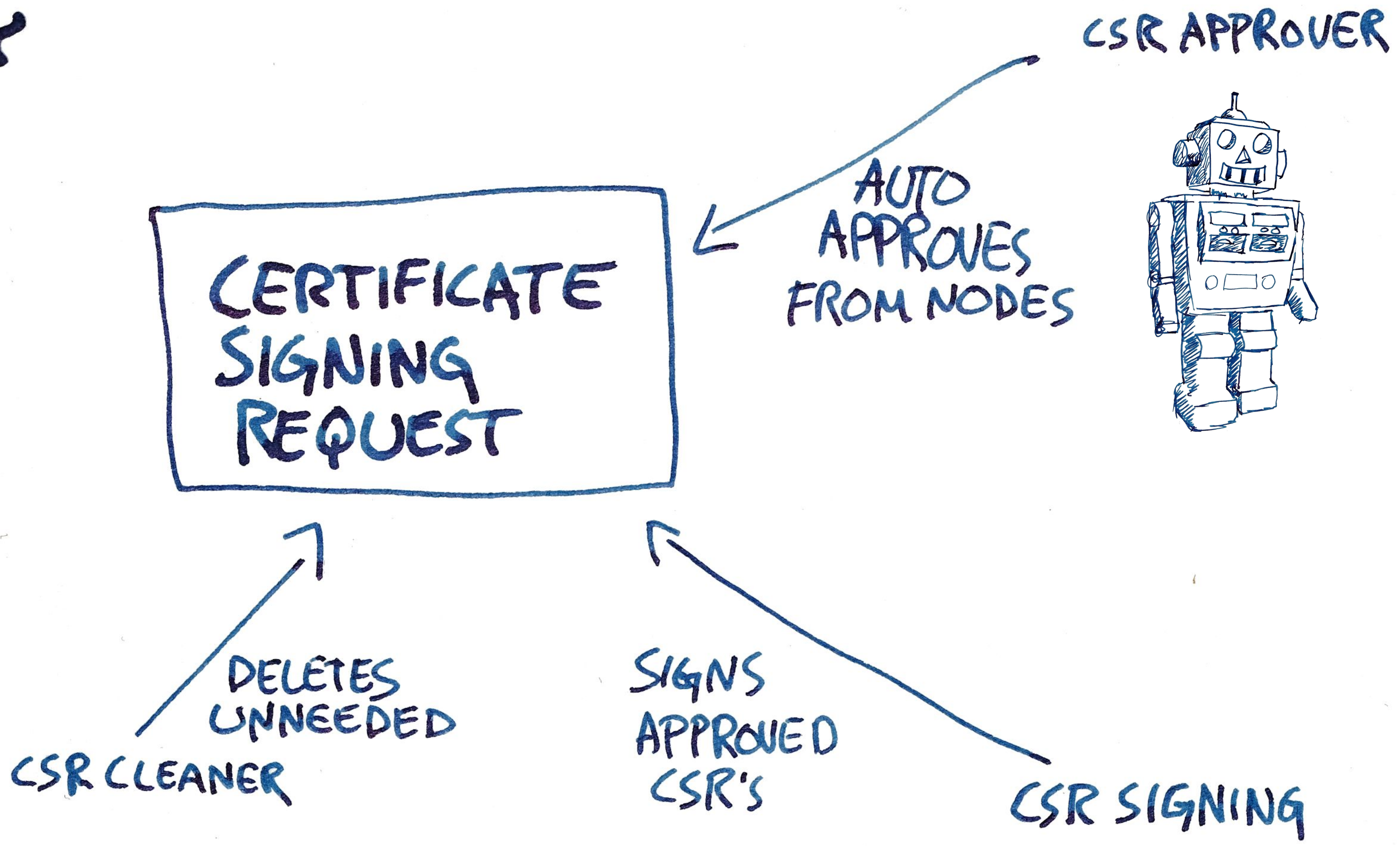
STANDING QUERY TABLE JOIN



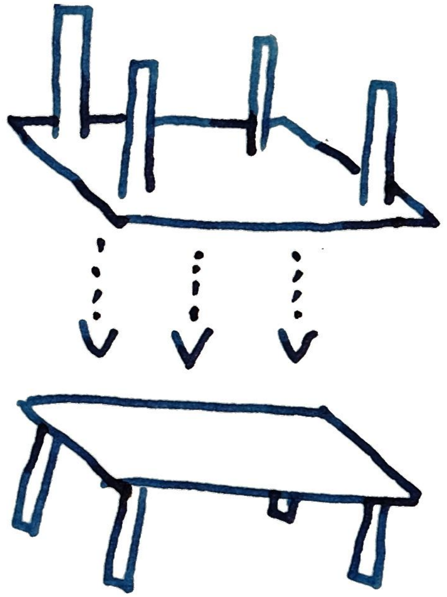


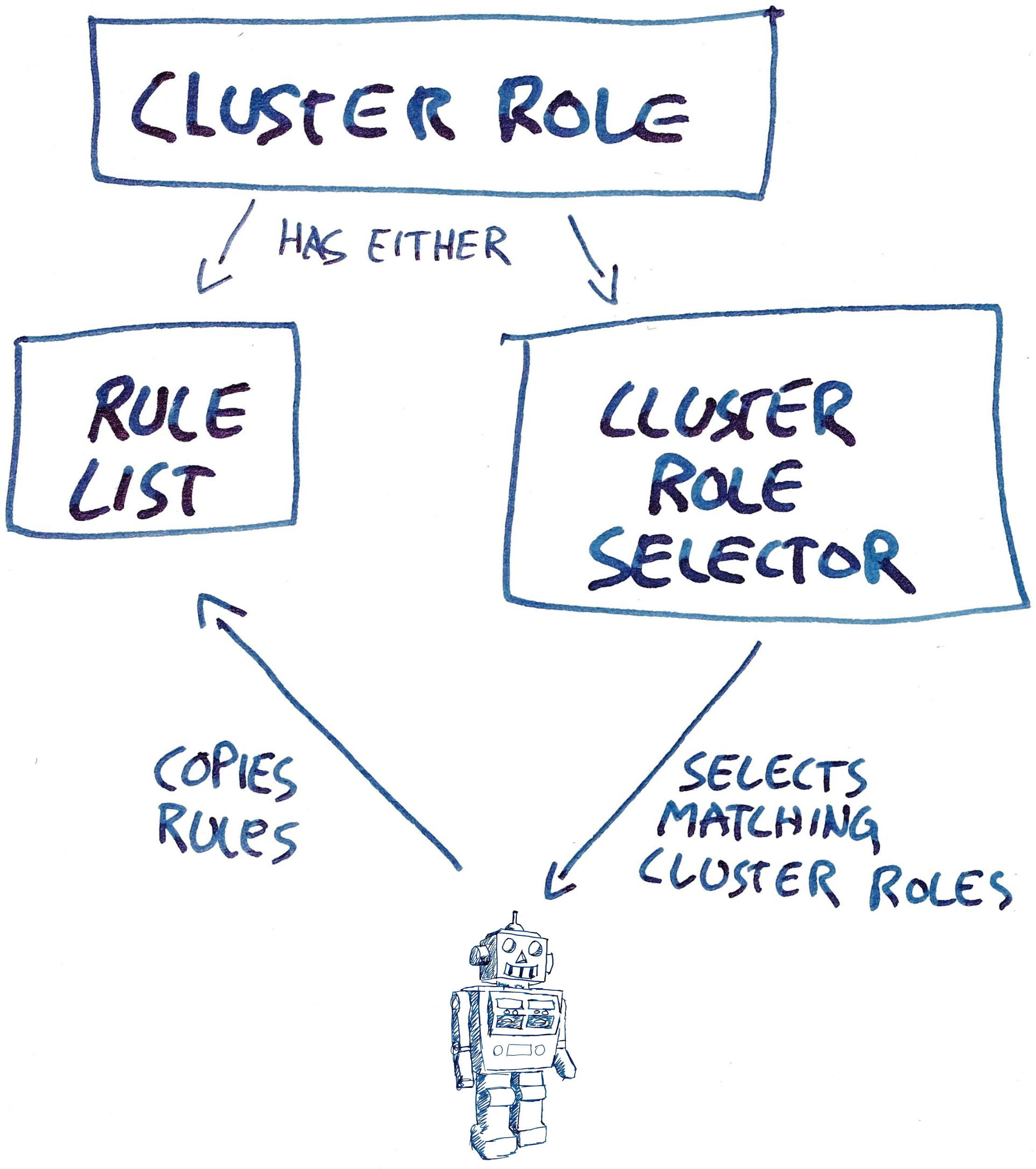
STANDING QUERY TABLE JOIN





STANDING QUERY TABLE JOIN





CLUSTER ROLE - AGGREGATION

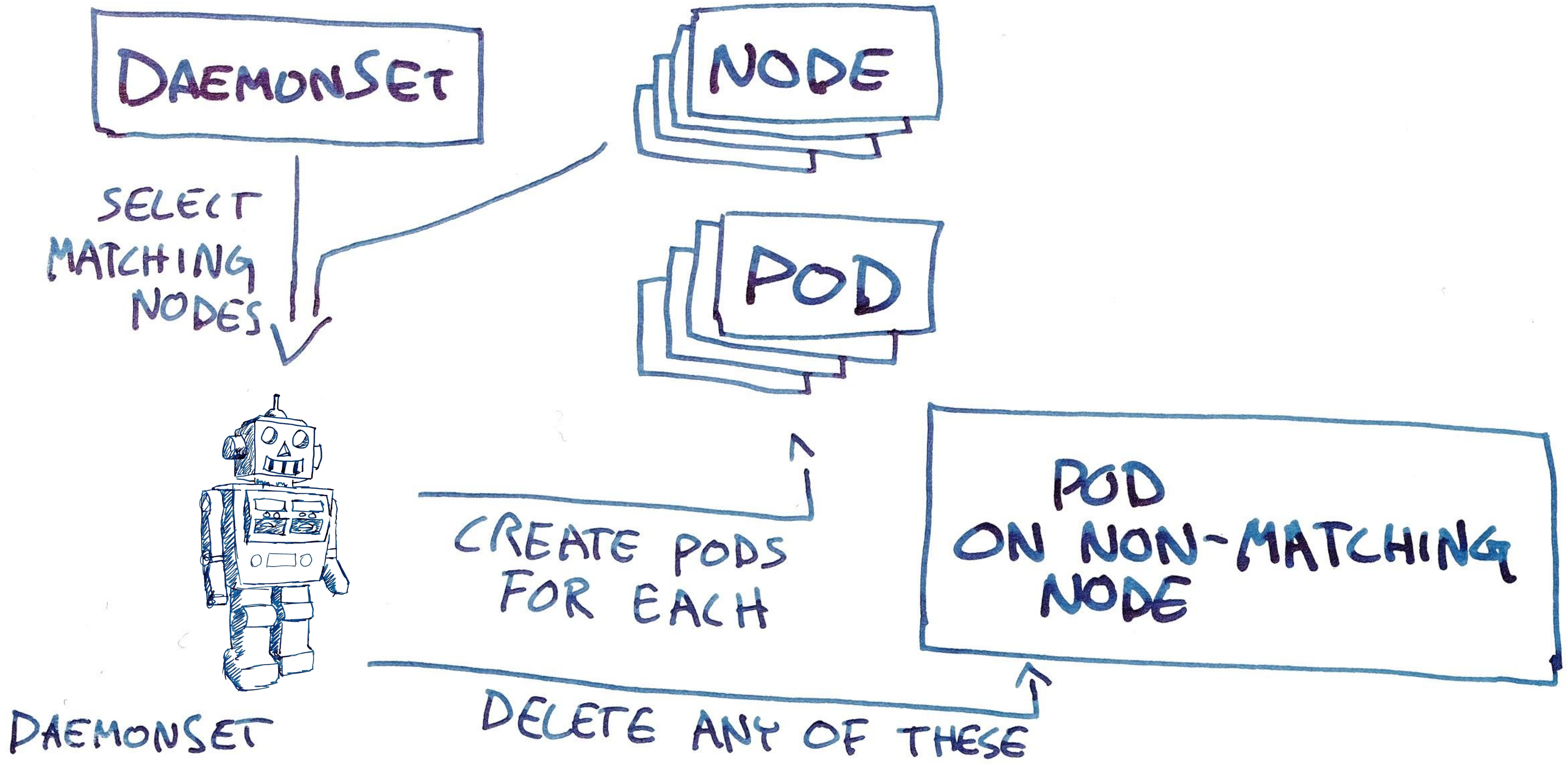
bi/injections

- Bijection maintainer (For X, do Y; for ~X, do ~Y)
  - DaemonSet, serviceaccount, persistentvolume-binder
- Injection maintainer / candy-wrapper
  - csrsigner, ttl, bootstrapsigner, nodeipam, nodelifecycle, root-ca-cert-publisher, cloud-node, scheduler
- calendar (injection with moon phases): scheduledjob
- babysitter (injection with complex thing): statefulset, kubelet
- oven (thing with complex injection): job
- external system integration (“operator” pattern):
  - service, route, cloud-node-lifecycle

$\overset{\text{IN}}{\text{BI}}$JECTION ENFORCER

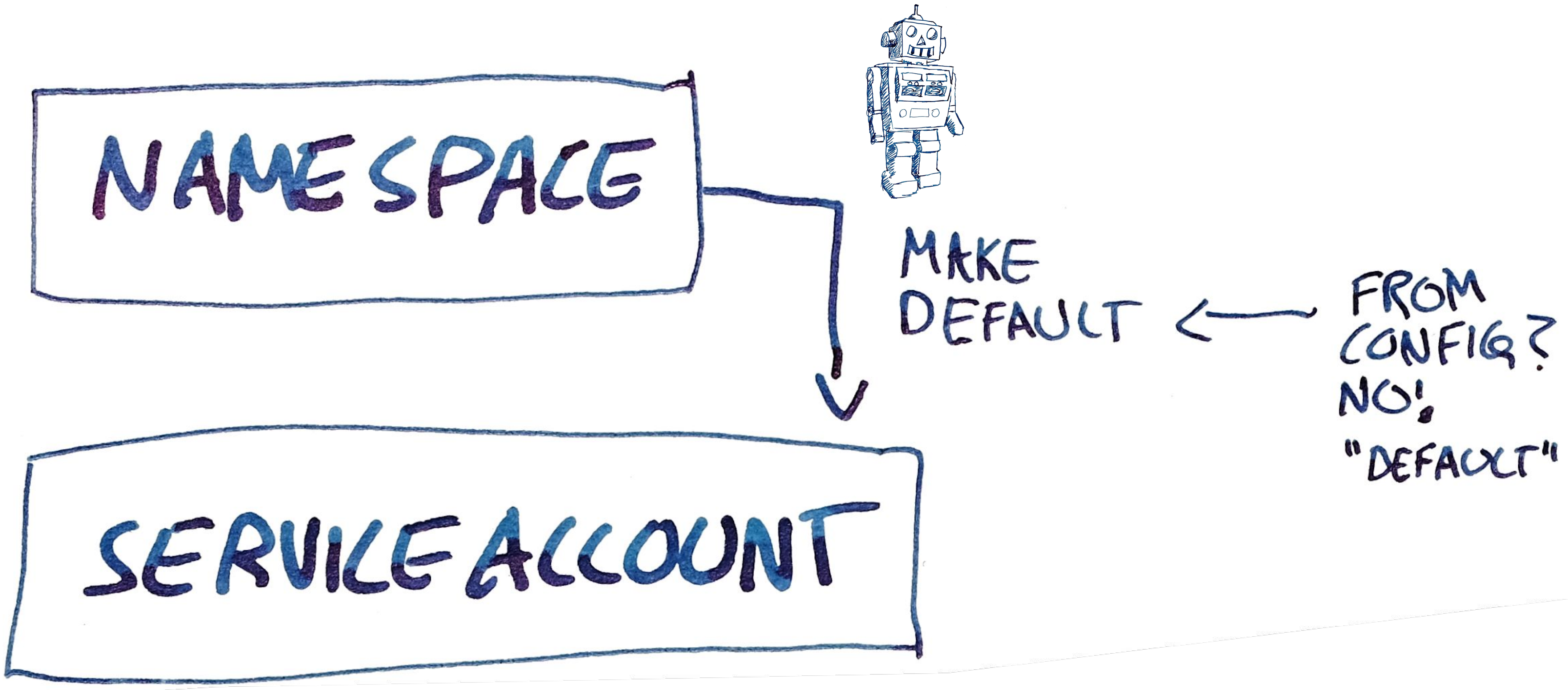
$$\begin{tikzcd}
1 \arrow[d, leftrightarrow] & 2 \arrow[d, Leftrightarrow] & 3 \arrow[d, Leftrightarrow] & 4 \arrow[d, Leftrightarrow] & \cdots \\
A & B & C & D & \cdots
\end{tikzcd}$$

$^{\text{IN}}_{\text{BI}}$JECTION ENFORCER

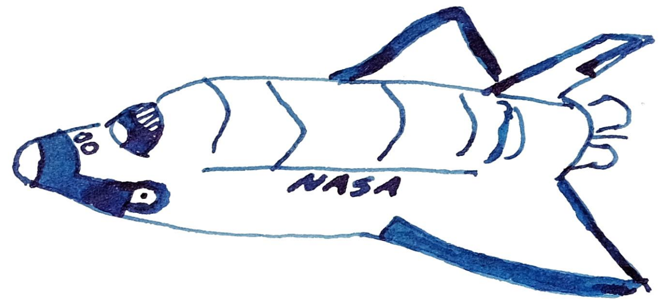


$^{\text{IN}}_{\text{BI}}$JECTION ENFORCER

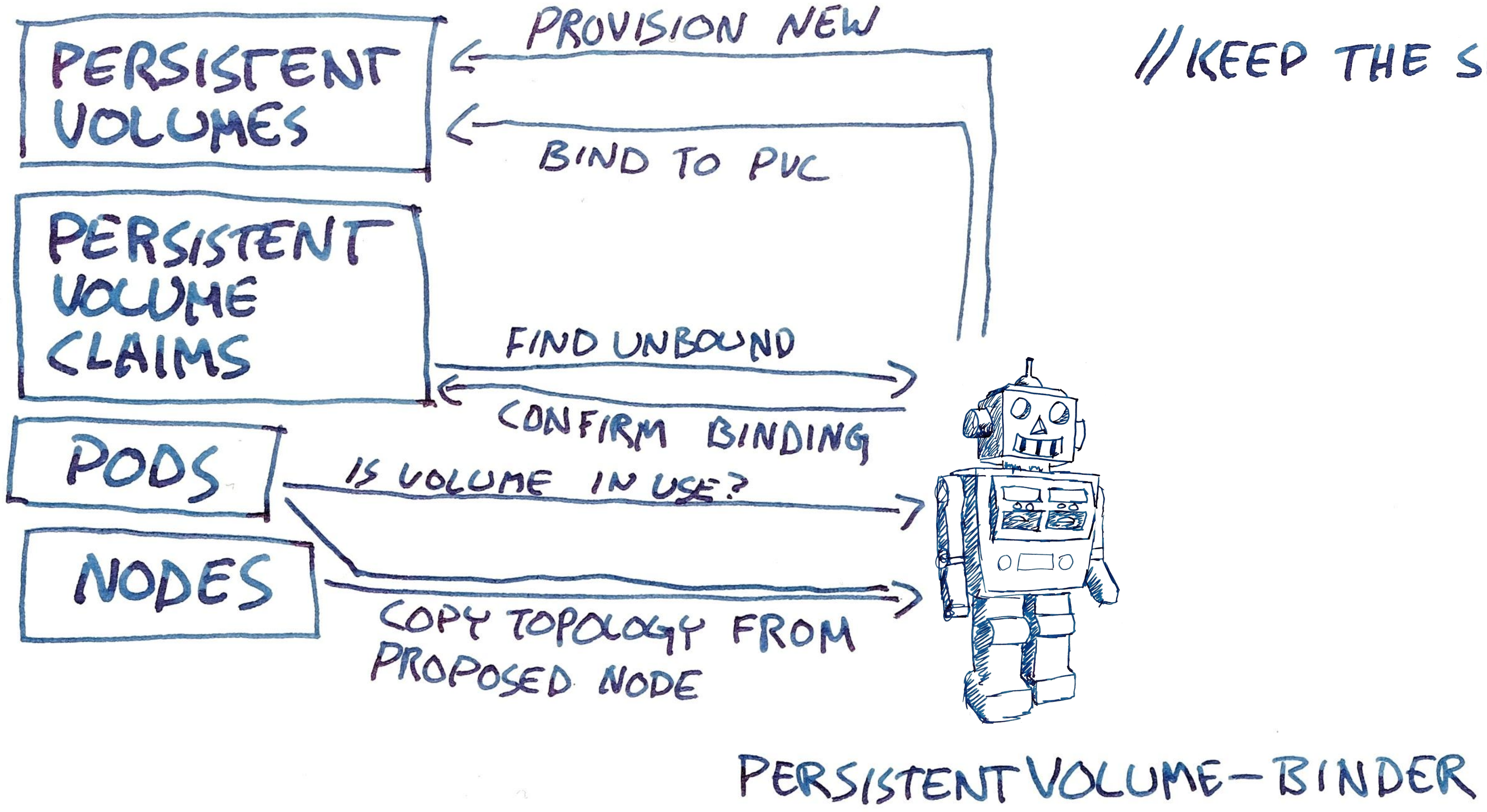
SERVICE ACCOUNT



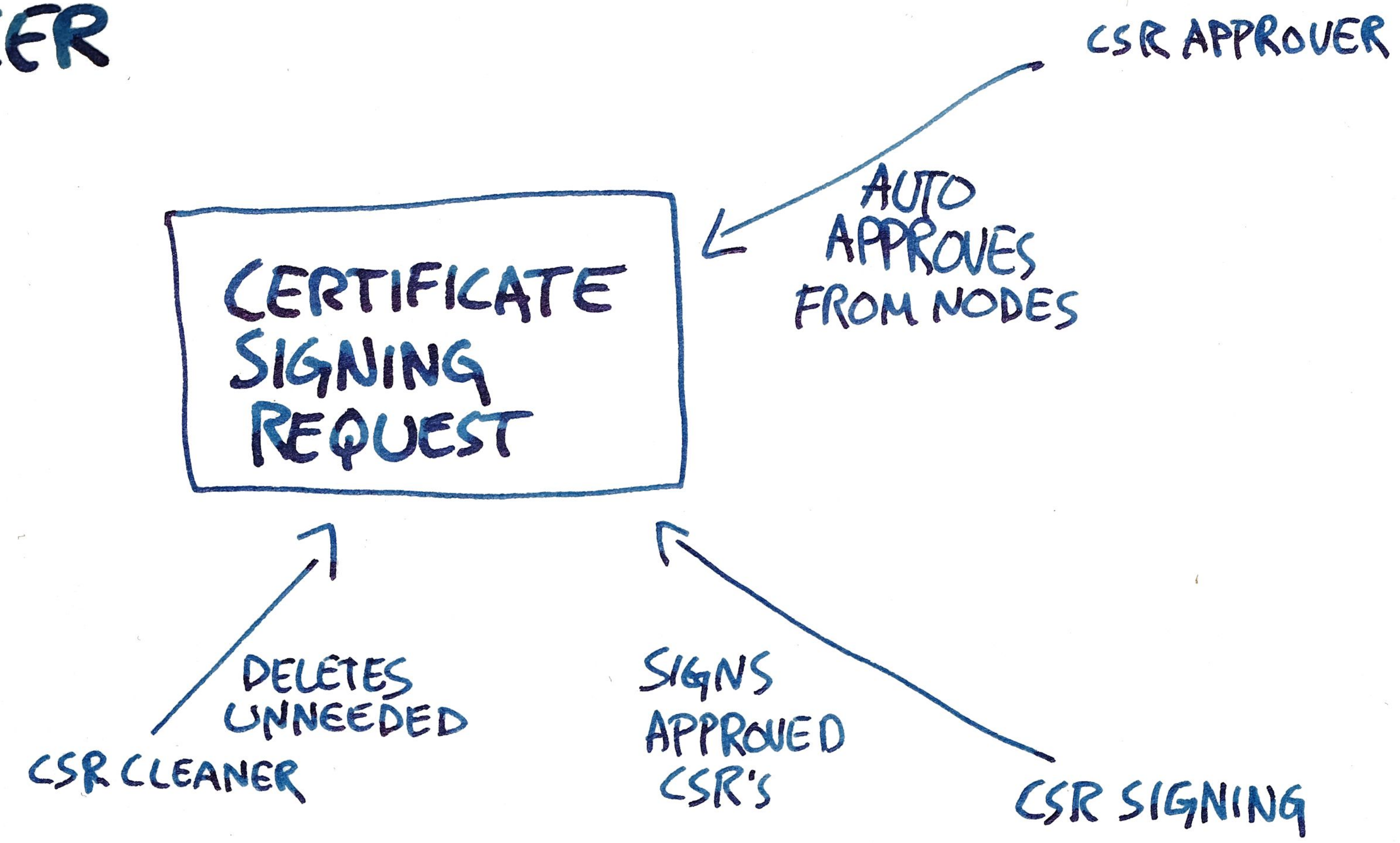
$^{IN}_{BI}$JECTION ENFORCER

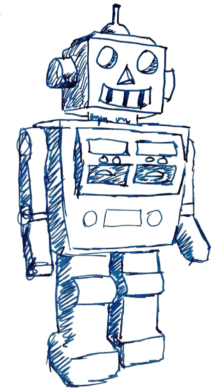


// KEEP THE SPACE SHUTTLE FLYING



$^{\text{IN}}_{\text{BI}}$JECTION ENFORCER

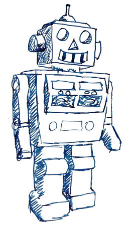




$^{\text{IN}}_{\text{BI}}$JECTION ENFORCER

NODE

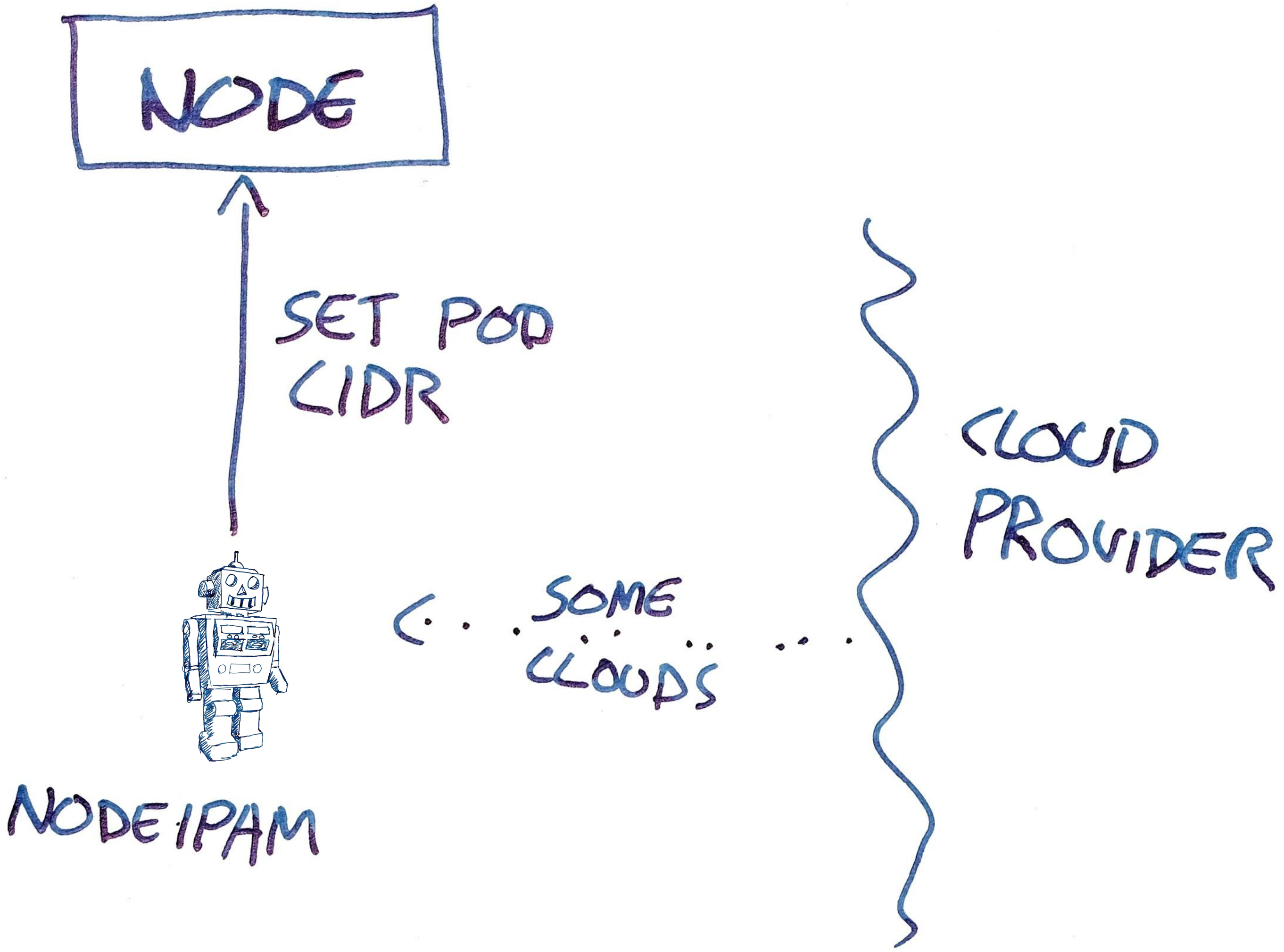
SET ANNOTATION
"NODE.ALPHA.KUBERNETES.IO/TTL"
BASED ON CLUSTER SIZE



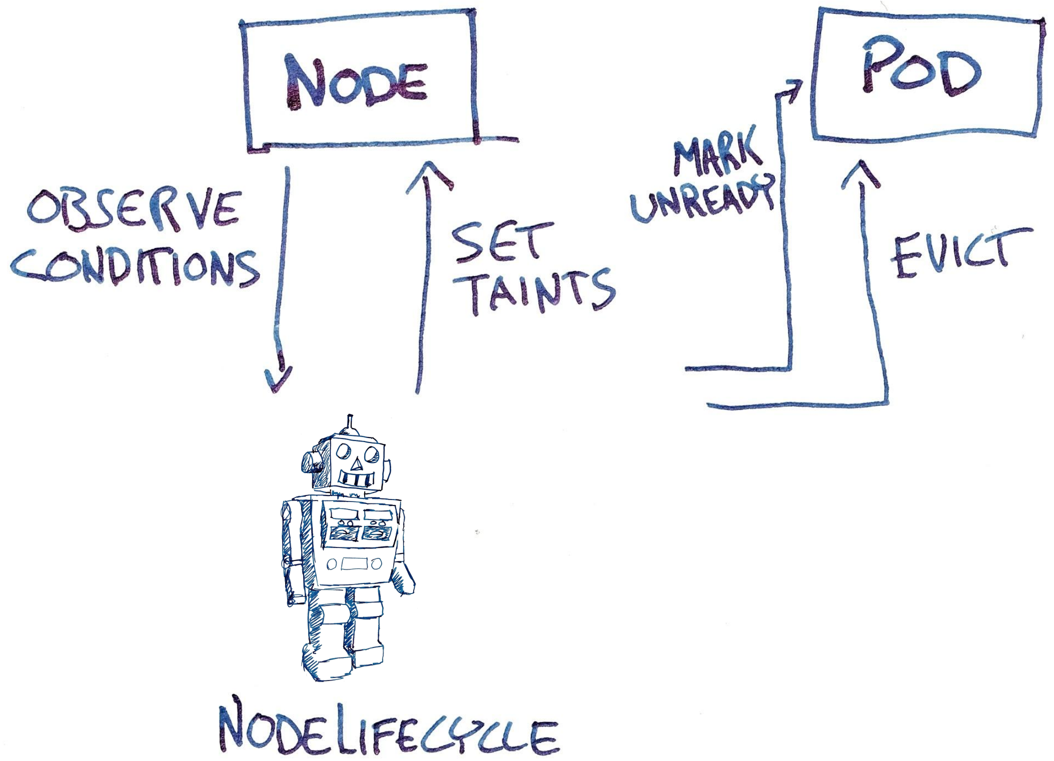
TTL CONTROLLER

MEANS CACHE TIME
FOR SECRETS, CONFIGMAPS, ...

~~IN~~ BIJECTION ENFORCER

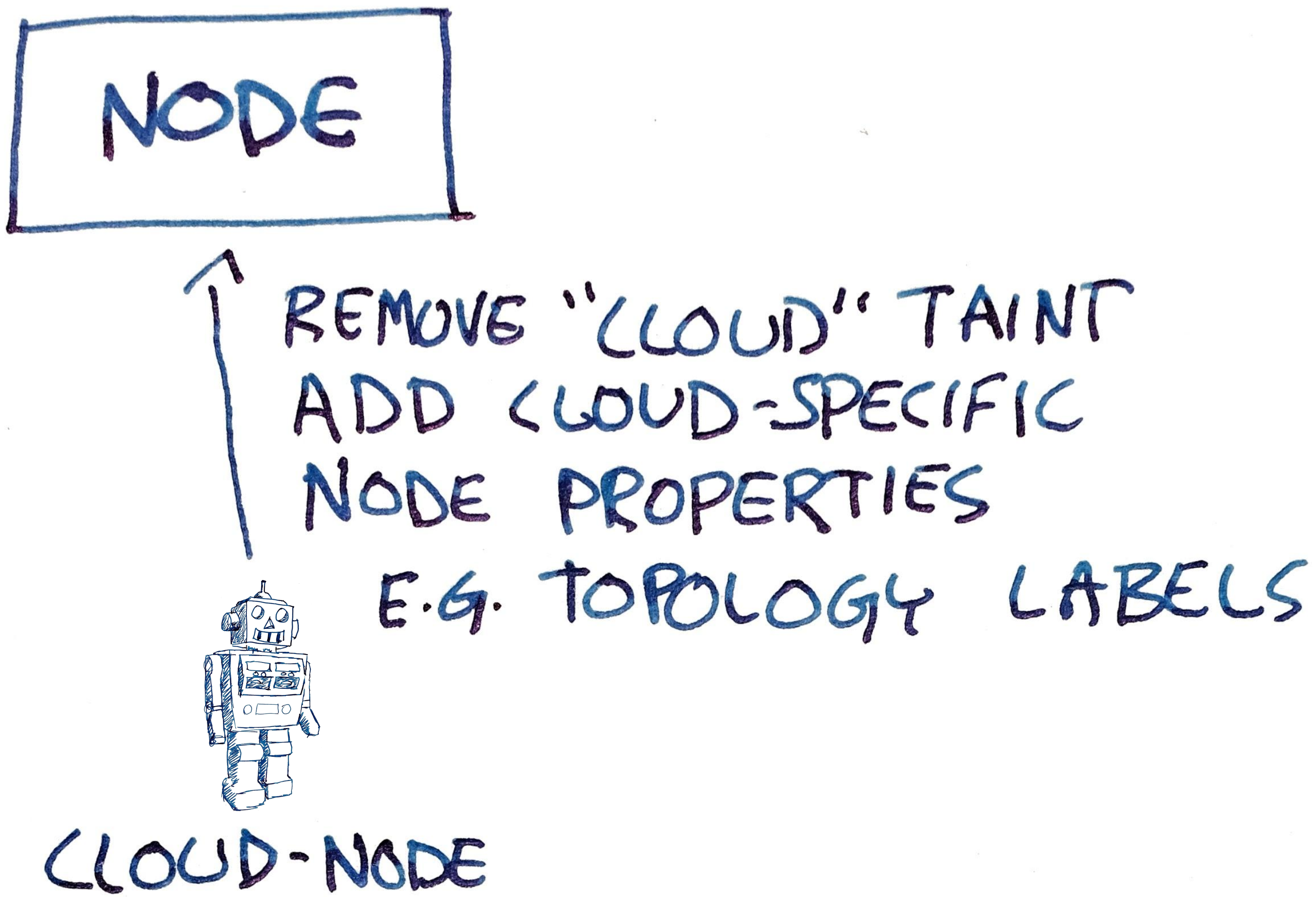


$^{IN}_{BI}$JECTION ENFORCER

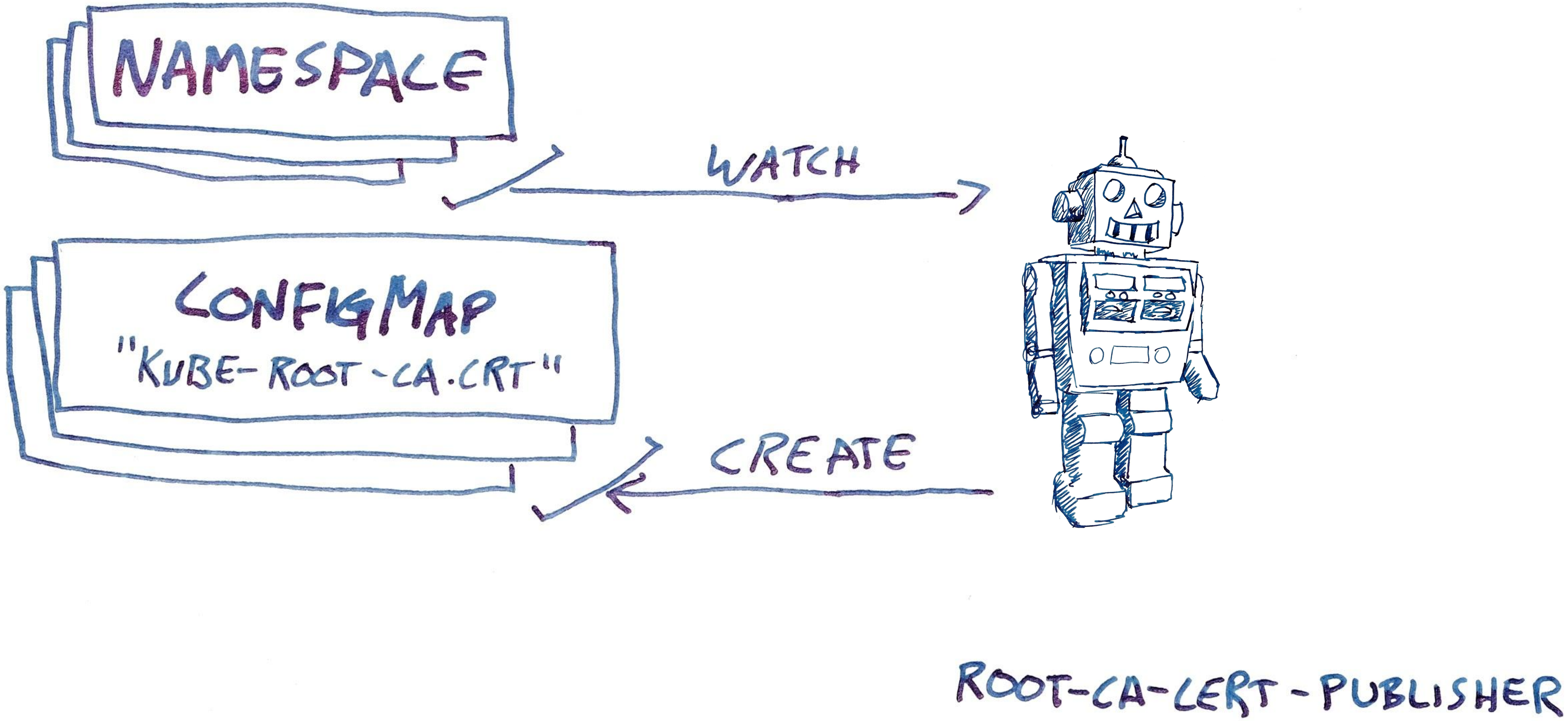


TAKE CHARGE OF THE K8S RESOURCES IF SOMETHING HAPPENS TO KUBELET

$\substack{\text{IN}\\ \text{BI}}$JECTION ENFORCER

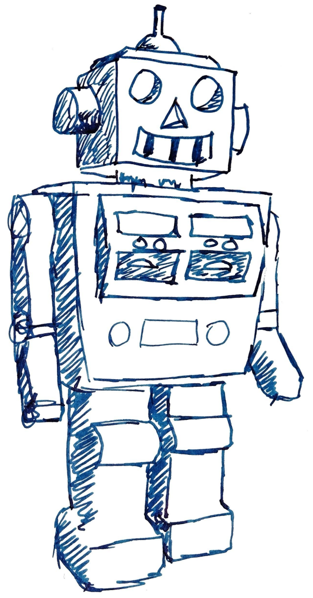


$^{\text{IN}}_{\text{BI}}$JECTION ENFORCER

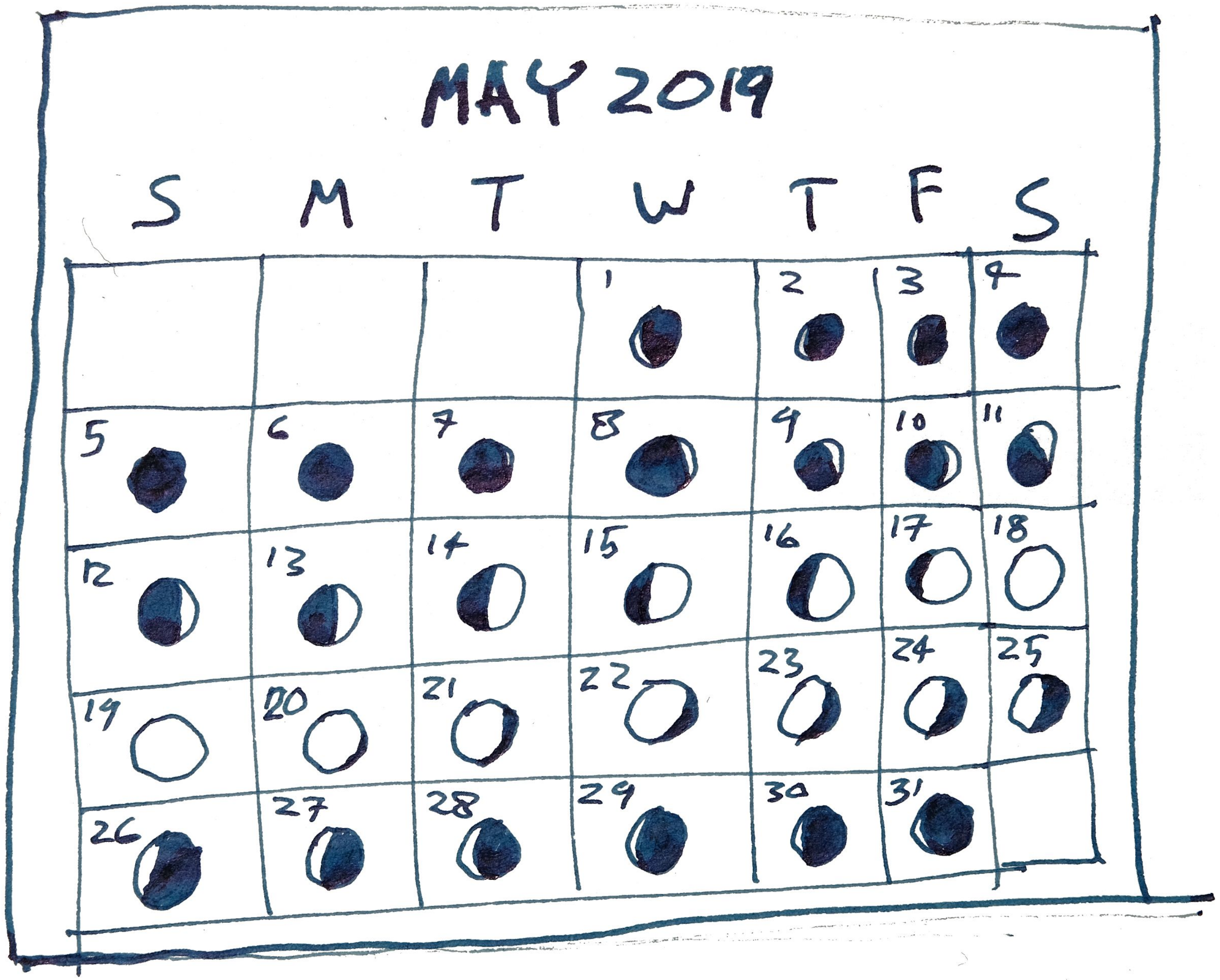


$\overset{\text{IN}}{\text{BI}}$JECTION ENFORCER

scheduler



~~IN~~ BIJECTION ENFORCER



scheduledjob

$\substack{\text{IN}\\\text{BI}}$JECTION ENFORCER

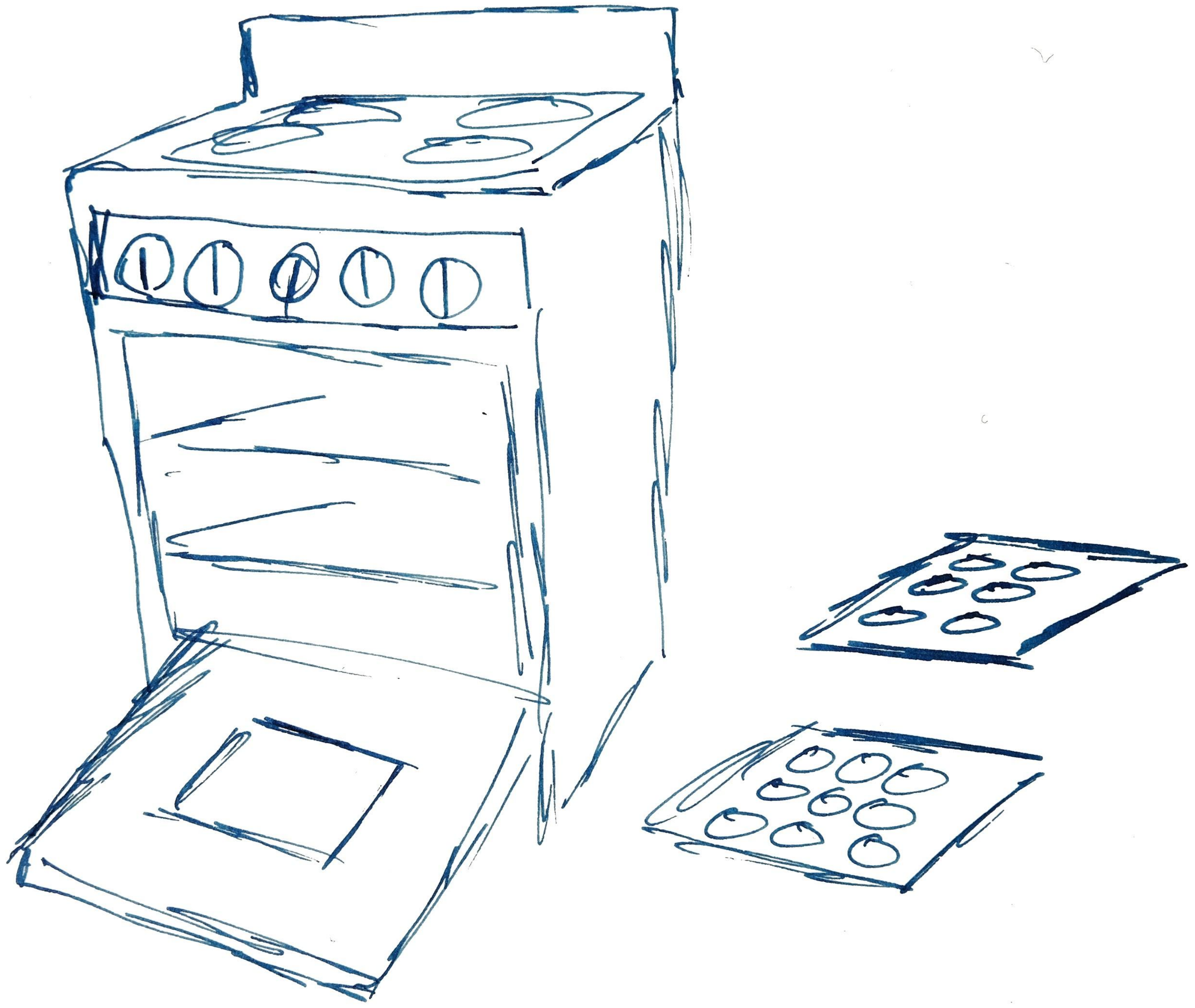
statefulset

$\overset{\text{IN}}{\text{BI}}$JECTION ENFORCER

kubelet

$\overset{\text{IN}}{\text{BI}}$JECTION ENFORCER

job



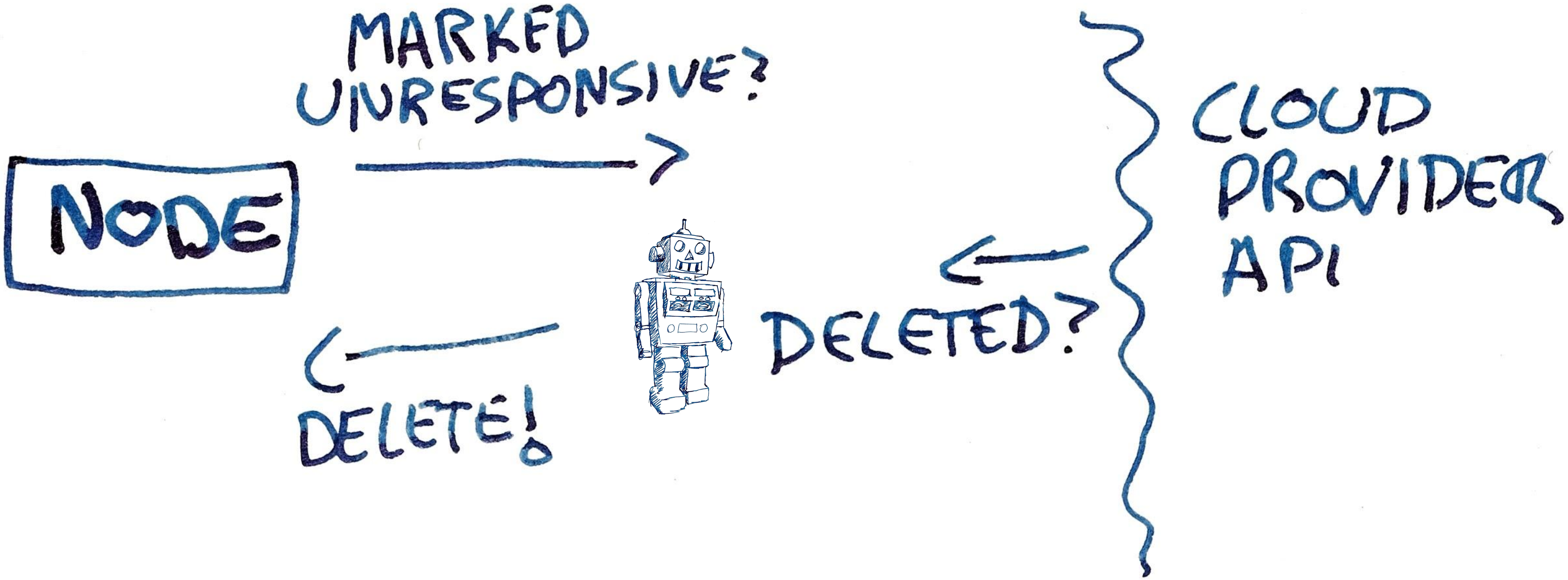
$\overset{\text{IN}}{\text{BI}}$JECTION ENFORCER

service

$\overset{\text{IN}}{\text{BI}}$JECTION ENFORCER

route

$^{\text{IN}}_{\text{BI}}$JECTION ENFORCER



CLOUD-NODE-LIFECYCLE

Thank you for reading my draft slides

I still have more work to do as you can see!

KubeCon | CloudNativeCon

Europe 2019