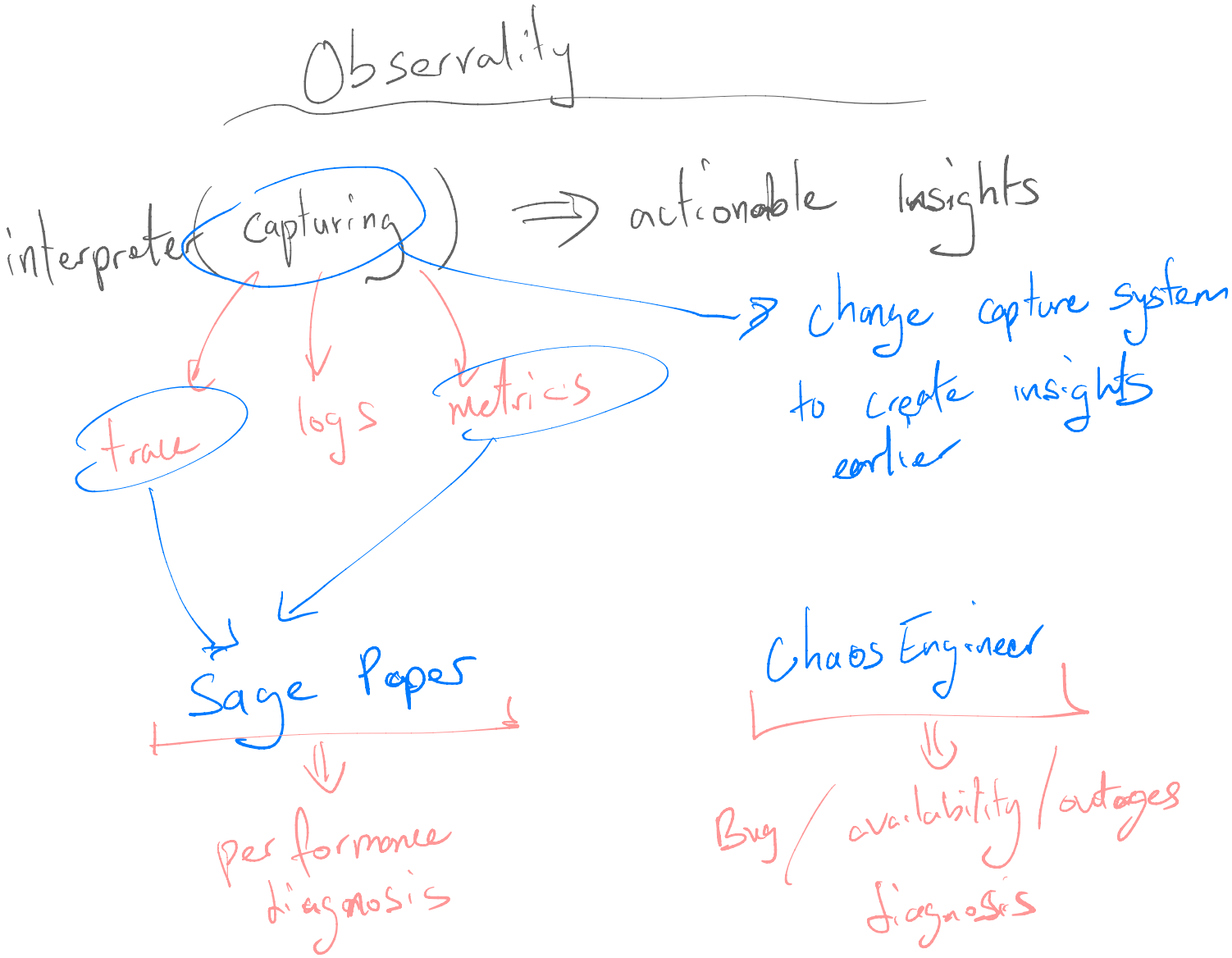
Episode 07


Observality
interpreter
capturing
actionable insights
change capture system to create insights earlier
traces
logs
metrics
Sage Paper
performance diagnosis
Chaos Engineer
Bug / availability / outages diagnosis

SRE / DevOps

shift in both code development & Infra management

Challenges / Cons

1. new types of errors / faults
2. Lack of visibility of entire system
3. scale ⇒ Many more components / error / faults
4. heterogeneity ⇒ many distinct / differents
5. Velocity ⇒ move fast / constant updates

Pro

1. automation ⇒ cloudification + softwarization
   a. automated provision
   b. automated generation of lifecycle events
2. excess of observability data "Open Telemetry"

AIOps

Artificial Intelligence for IT Operations

ML / DL / statistical technique

on

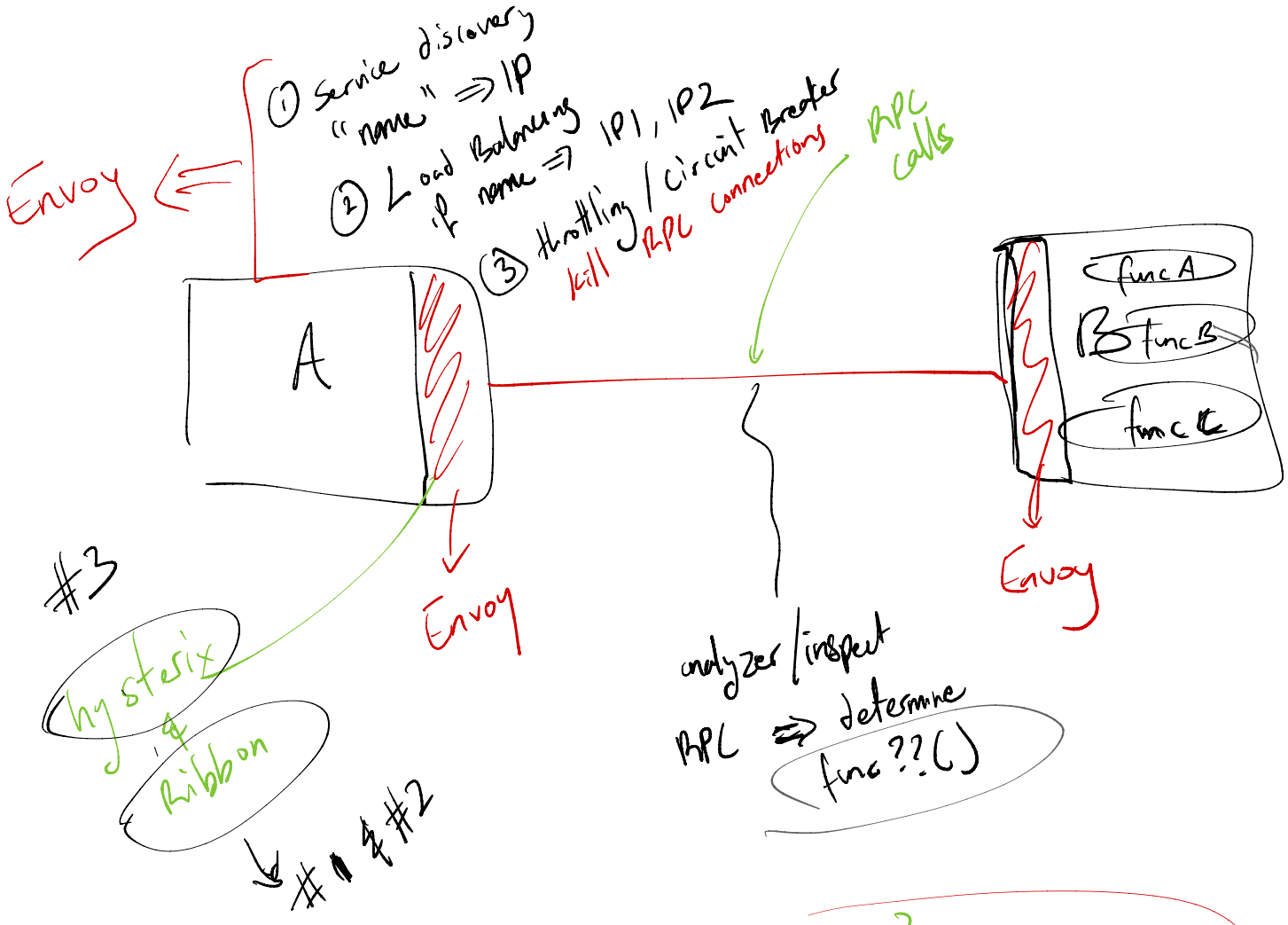
observability + lifecycle events

and fixing problems using kubernetes automation

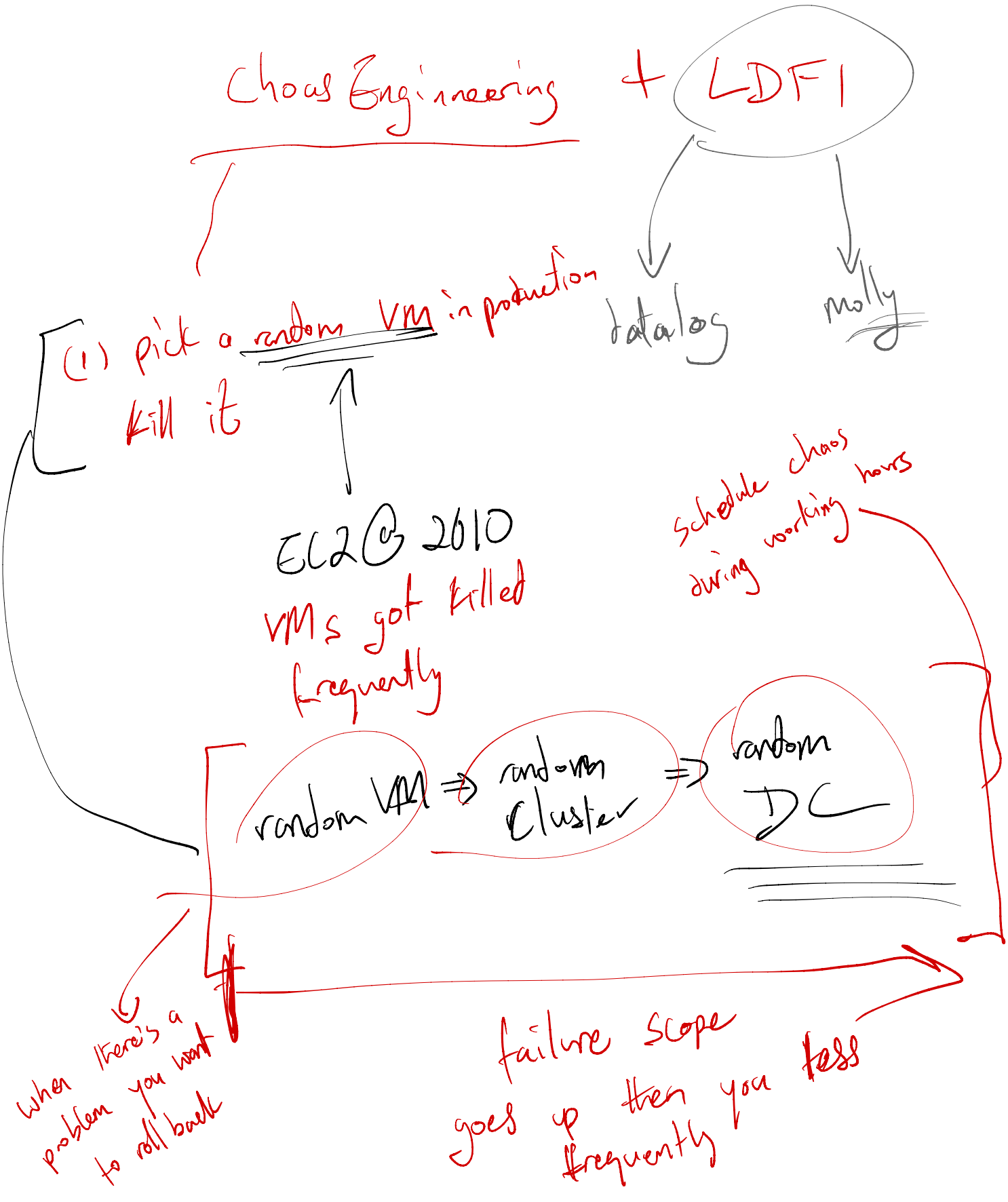
Unexpected Edge Cases (fire drill)

purpose of fire drills:

- quantify preparedness under emergency without the emergency
- educate people on procedures for handling emergencies



Coverage
= explore all
code paths


Chaos Engineering
+
LDFI
datalog
molly
(1) pick a random VM in production
kill it
EC2 @ 2010
VMs got killed
frequently
schedule chaos
during working hours
random VM
random
cluster
random
DC
when there's a
problem you want
to roll back
failure scope
goes up then you
frequently
less

Grelim / ChaosMesh

from B's perspective every failure

@ A shows up as a RPC issue

(*) n/w ⇒ high latency or failure

(*) server ⇒ high latency or failure

① delay pkts

② drops pkt

③ speedup pkts

④ slowdown pkts

Ideal Chaos Engineer algo = maximize coverage (# of code paths) + while minimize # of failures injected (failure impact production performance)

Sage

Velocity ⇒ constantly building new algo ⇒ ML but to incremental build model

⇓

different parts of graph model map to different servic

scale/heterogeneity ⇒ ML ⇒ supervised

supervised

⇓

labeled data

need a person to label it

a lot needs to be labeled & may need relabeling

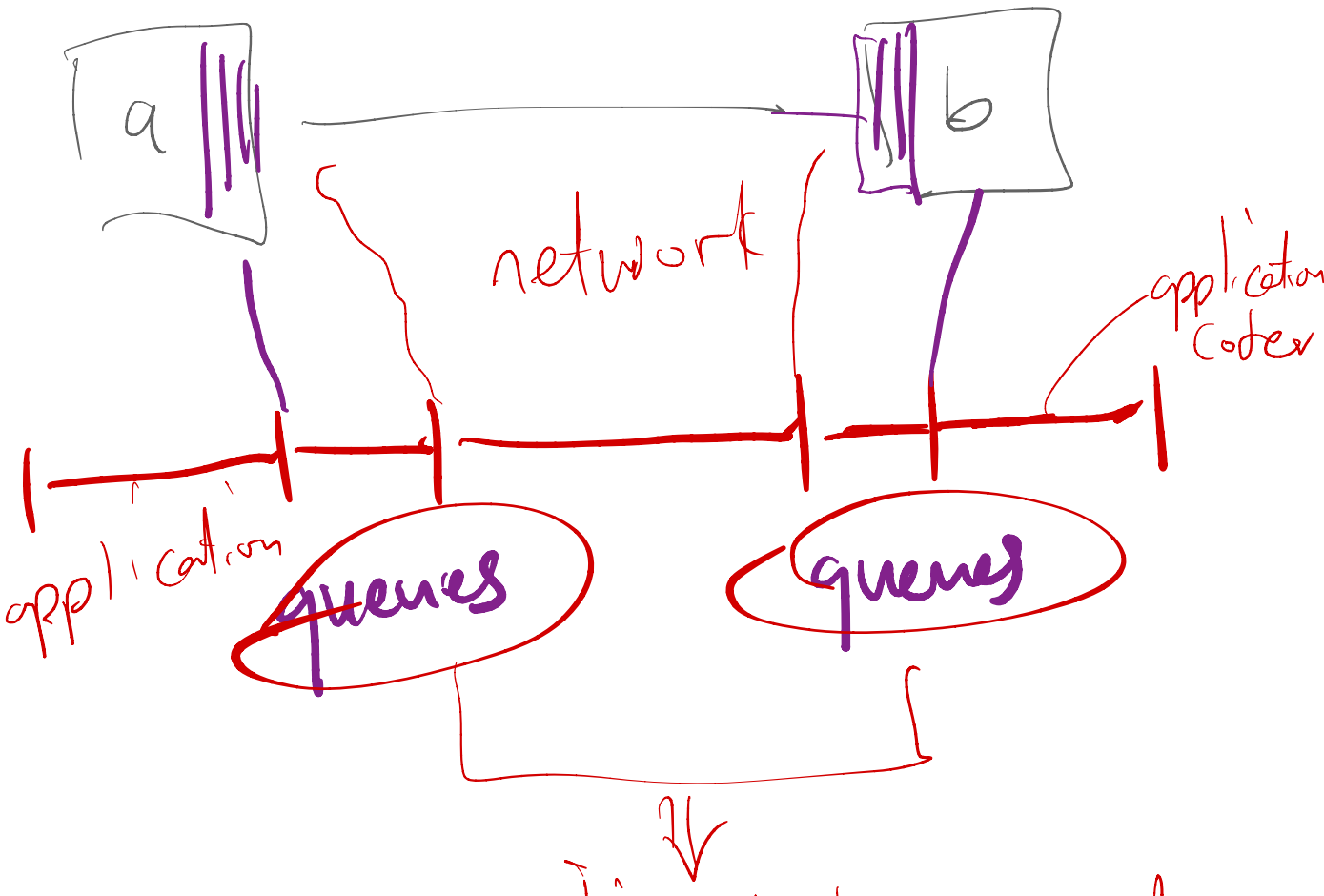
may have never seen the problem before

unsupervised

⇓

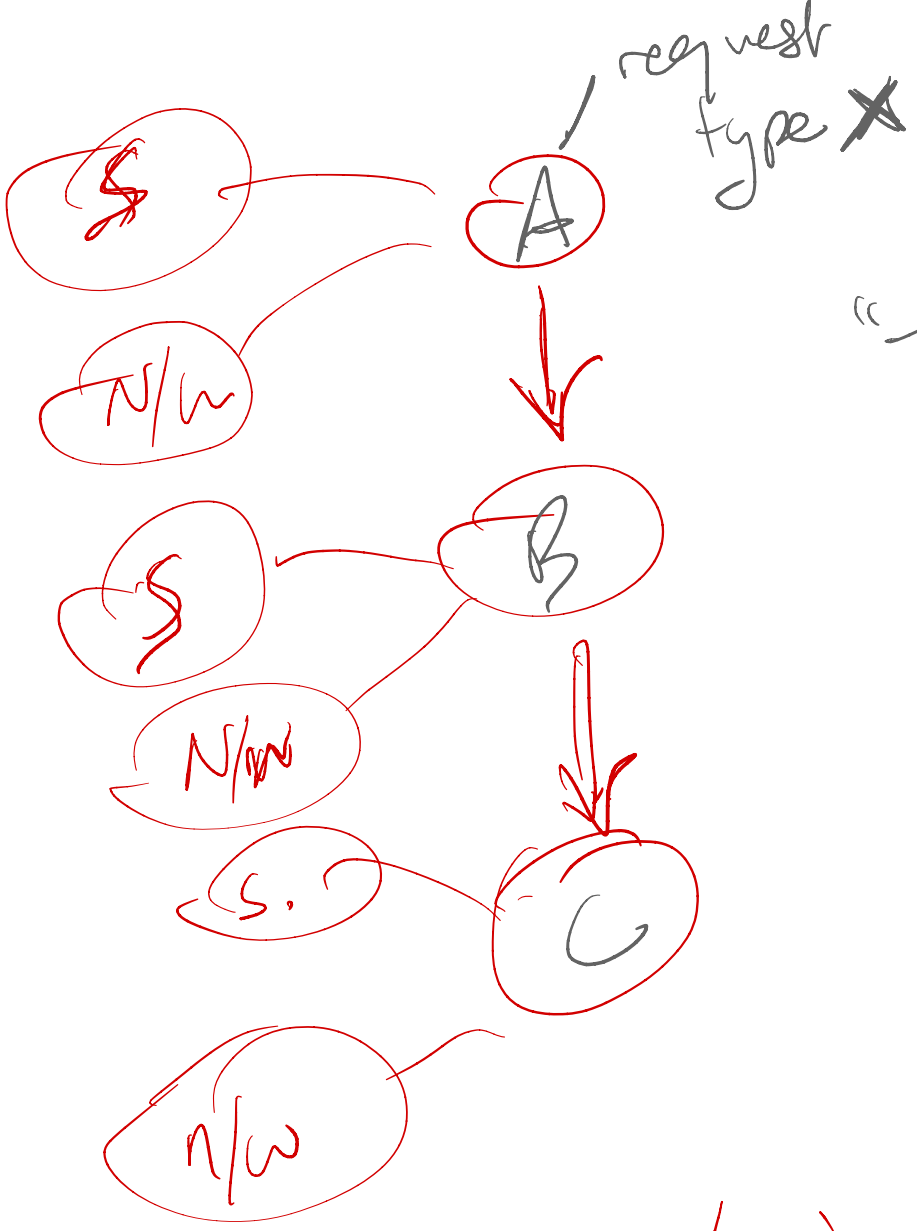
no labels for data

design mechanism which works without labeled data



waiting to be processed by the app

contention for CPU/mem which will increase wait time

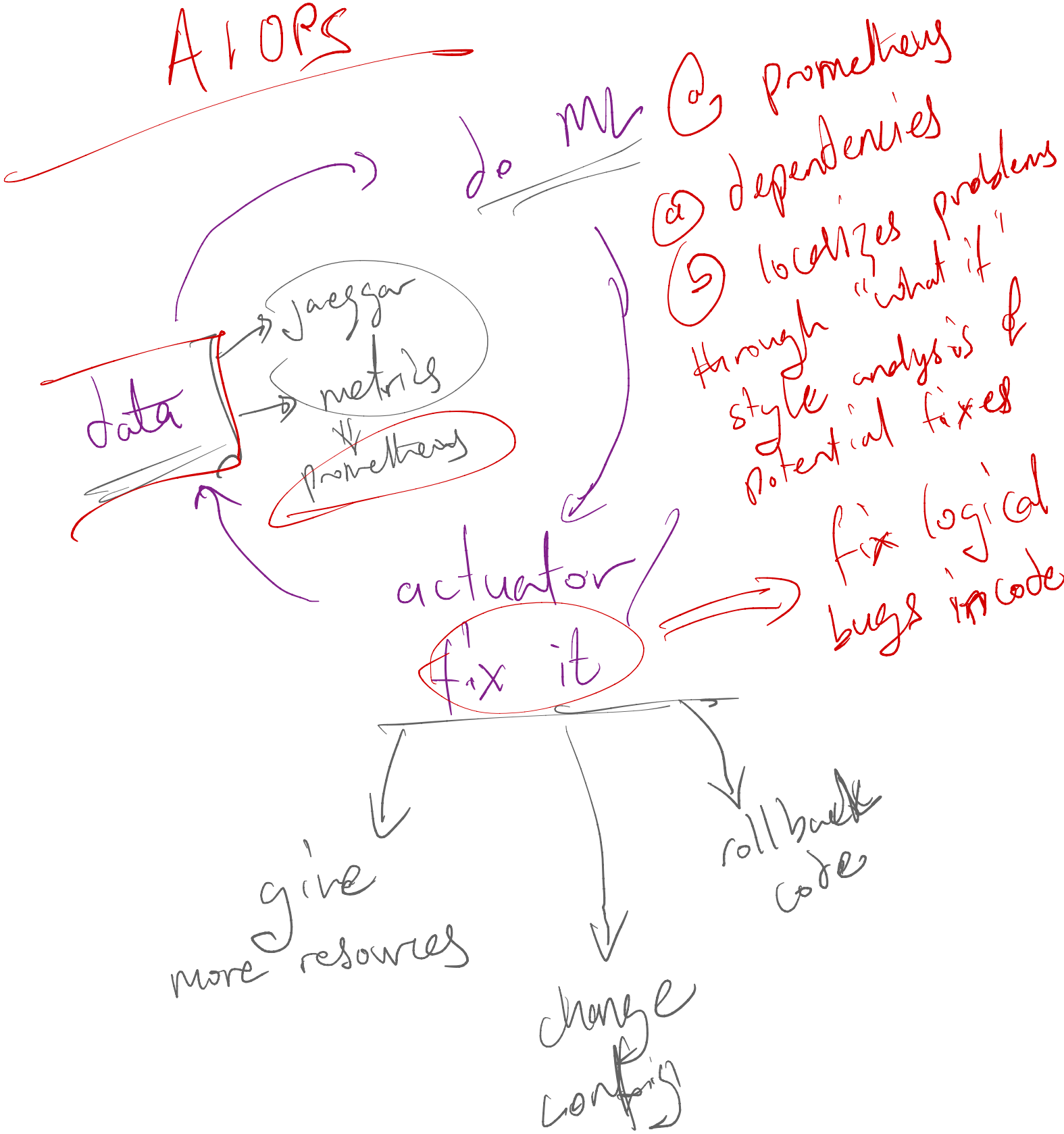


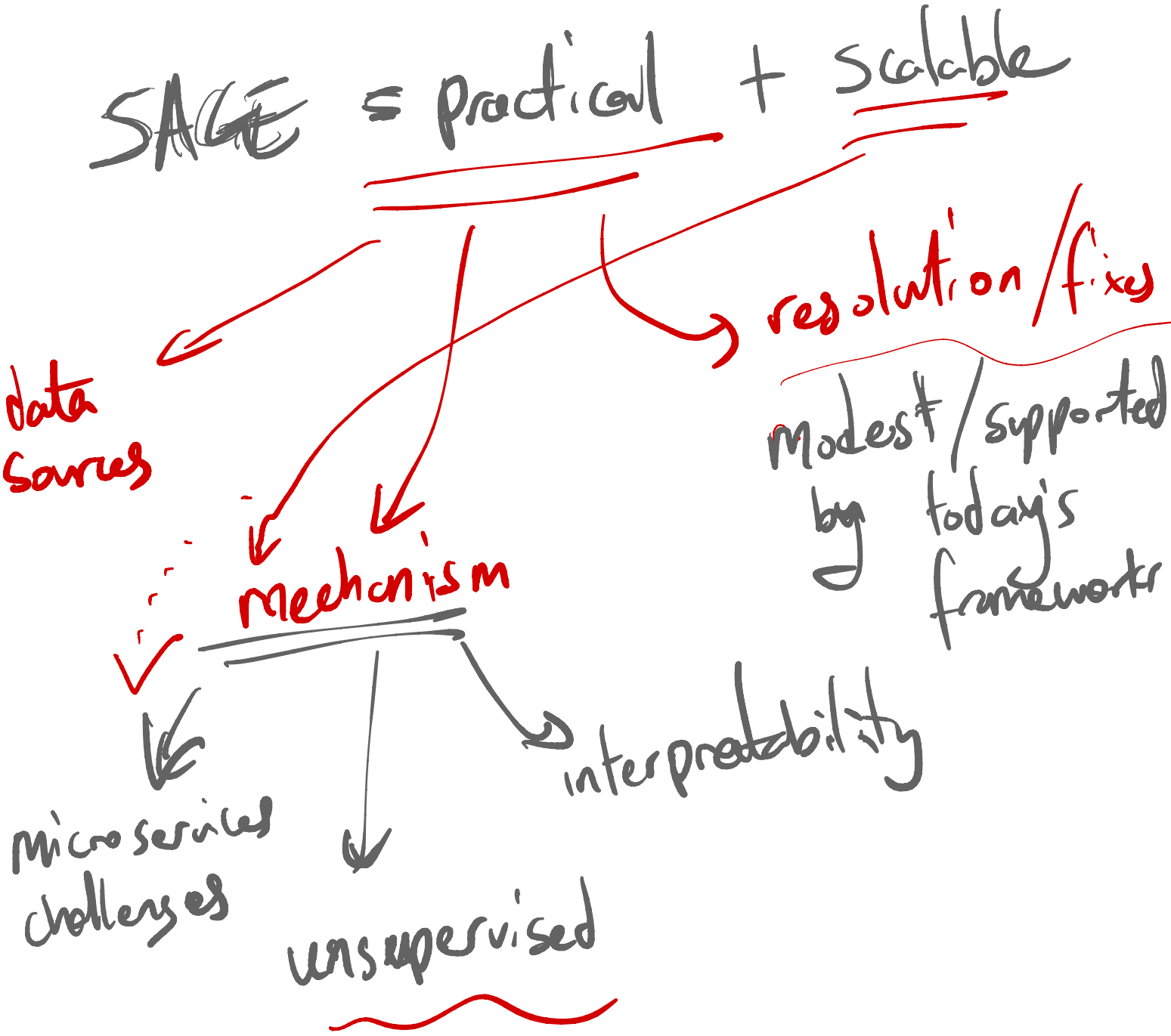
identify cause ⇒ rollbacking / fixing individual components

⇓

history of differences ⇒

You can check to see if there's a trace that corresponds to what if


AIOPS
do ML
Prometheus
dependencies
localizes problems through "what if" style analysis & potential fixes
fix logical bugs in code
Jaegar
metrics
Prometheus
data
actuator
fix it
give more resources
change config
rollback code


SAGE = practical + scalable
data sources
mechanism
resolution/fixes
modest/supported by today's frameworks
microservices challenges
unsupervised
interpretability

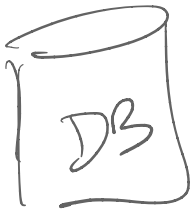
Deathstar Bench

* Benchmark of 5 apps written as Microservices

* largest corpus for evolution

* not representative

AIOps = KubeOps

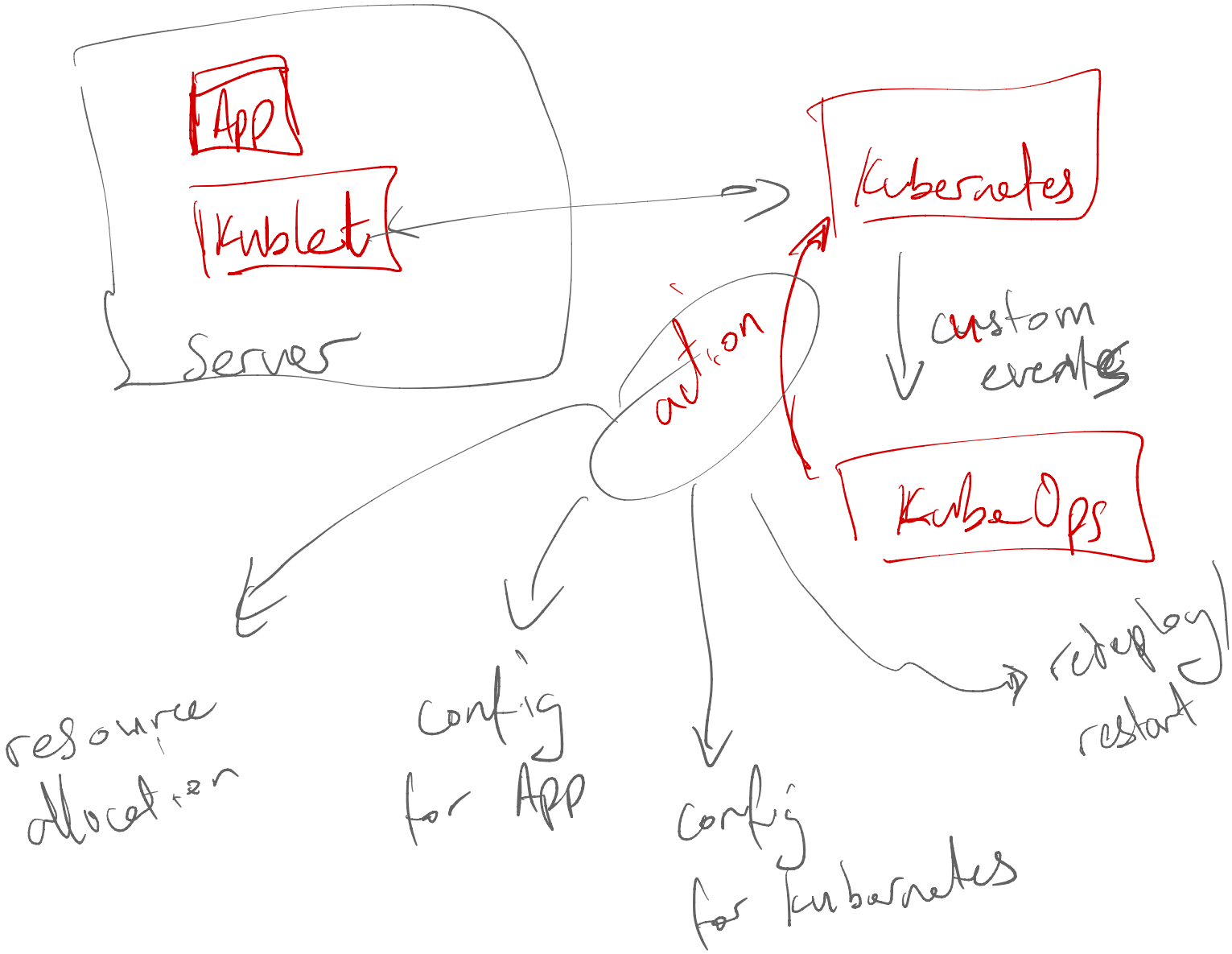


= DBAs

= special role for Admin Hadoop

specialized roles for administering one application

KubeOps = scripts that automate the jobs of this Admin


App
Kublet
Server
Kubernetes
custom
Kube Ops
resource
config
for App
config
for kubernetes
redeploy/
restart

Today

Chaos Engineering

* ideal algo.
* what it is
* Netflix

AIOps

* operational challenges / benefits of Microservice Ecosystem
* two approaches to AIOps
  * KubeOps
  * perf control loop (SAGE)
* observability for diagnosis

Next class (week after next)

* Config : auto config
* self driving Databases
* self driving Microservices