KubeCon | CloudNativeCon

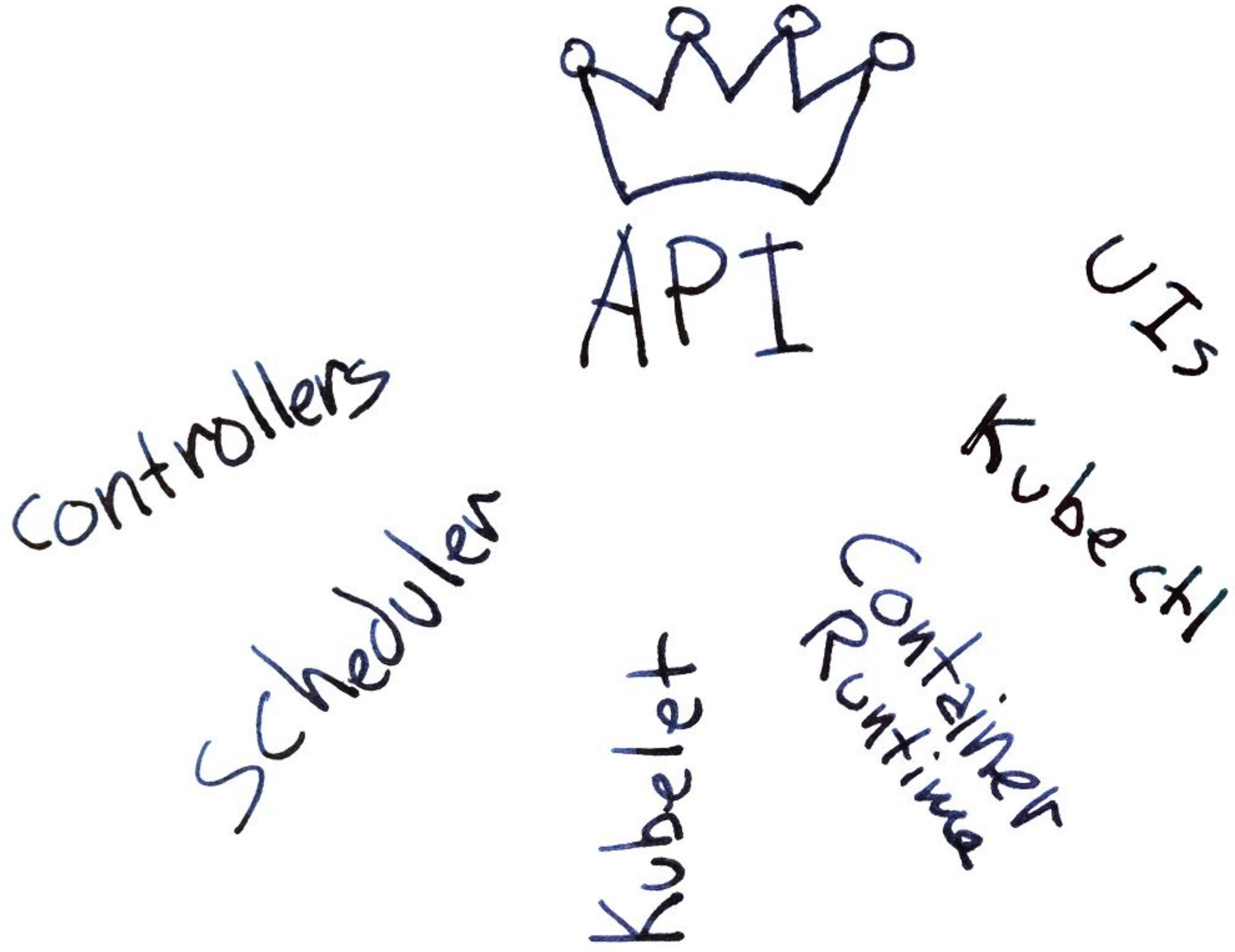
Europe 2018

Kubernetes-style APIs of the Future

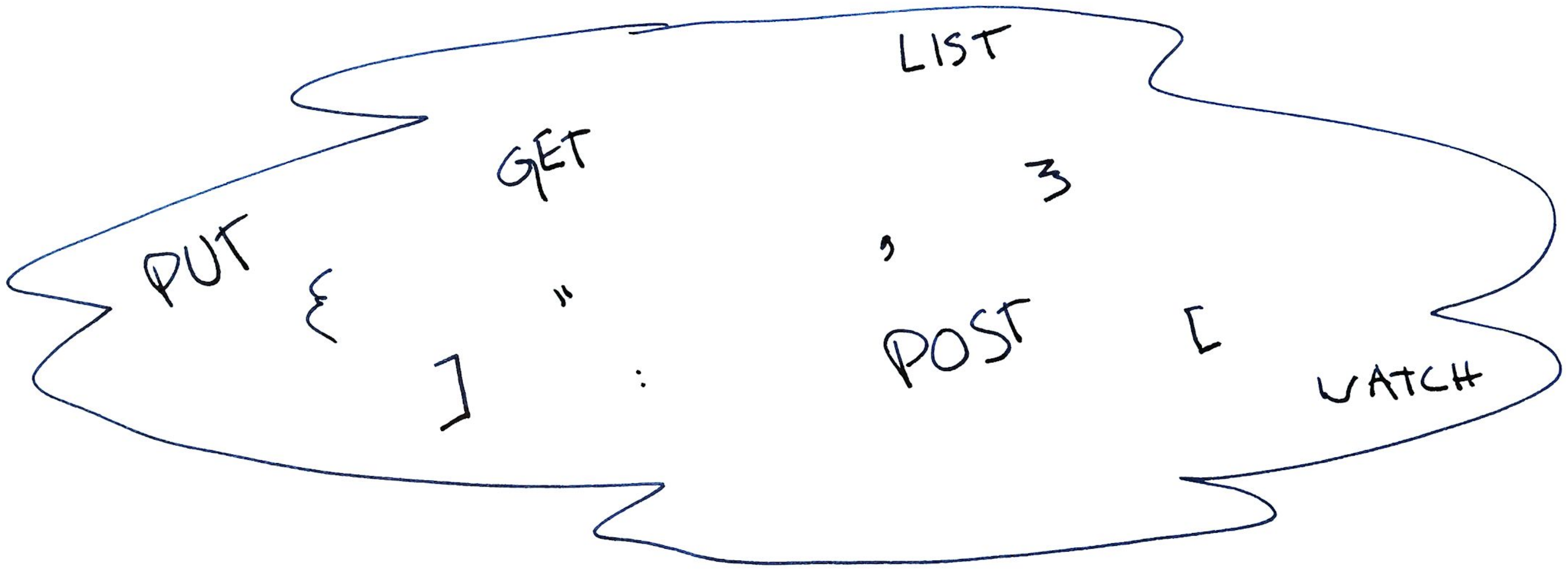
The Kubernetes Resource Model is coming to an API near you.

Daniel Smith
dbsmith@google.com
@lavalamp (github)
@originalavalamp (twitter)

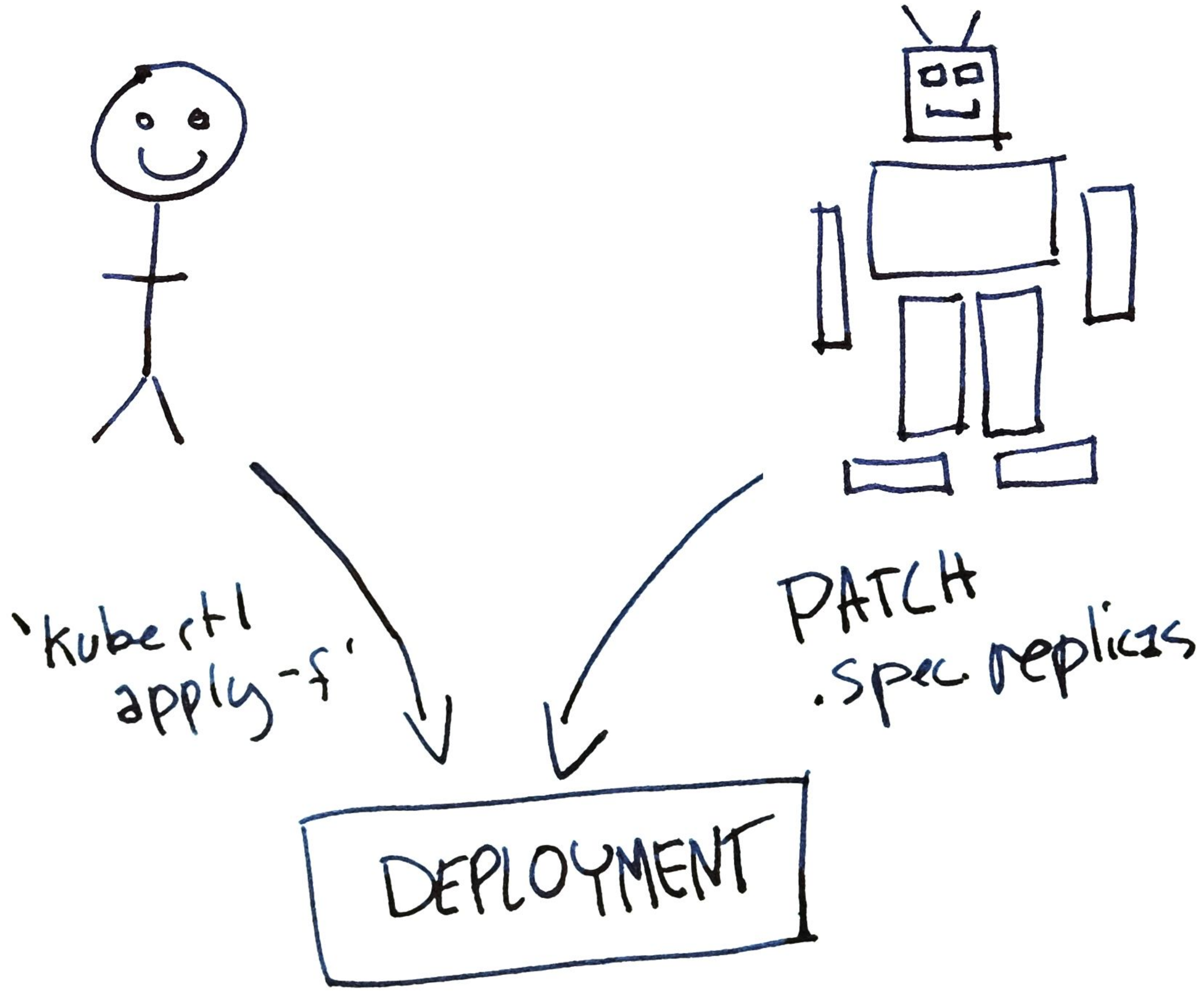
From the beginning, Kubernetes was API focused.



Kubernetes APIs evolved from a primordial soup of RESTy JSON.



People and automated systems used the API together.



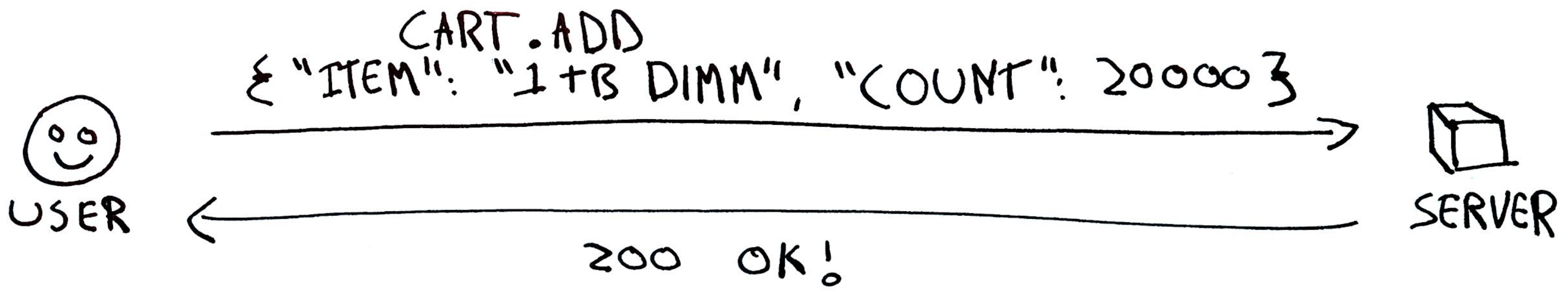
We built an API platform.

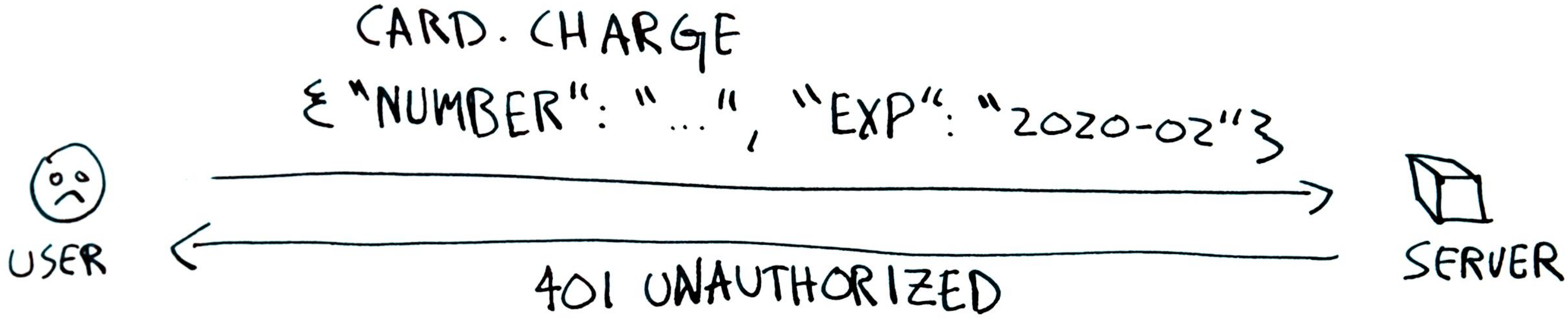
We built an API platform.
Oops?

We built an API platform.
Not a mistake.

We needed different things from our APIs.

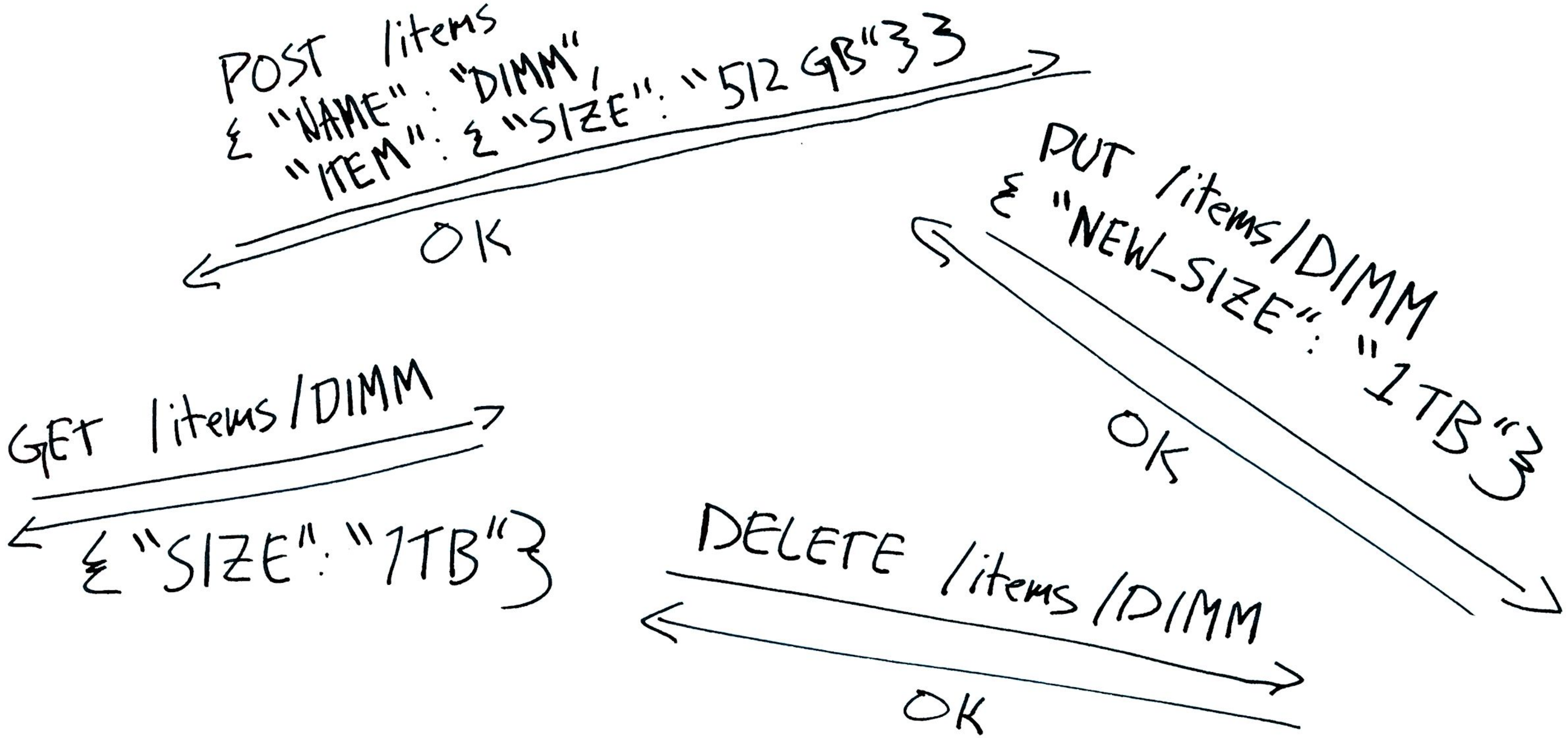
A brief, cherry-picked history of APIs


CART.ADD
{"ITEM": "1TB DIMM", "COUNT": 20000}
USER
SERVER
200 OK!


CARD.CHARGE
{"NUMBER": "...", "EXP": "2020-02"}
USER
SERVER
401 UNAUTHORIZED

What if the API is about the existence or state of some resource?

CRUD & REST


POST /items
{ "NAME" : "DIMM",
"ITEM" : { "SIZE" : "512 GB" } }
OK
PUT /items/DIMM
{ "NEW_SIZE" : "1 TB" }
OK
GET /items/DIMM
{ "SIZE" : "1TB" }
DELETE /items/DIMM
OK

We have the RPC version of object oriented programming. But something still seems off.

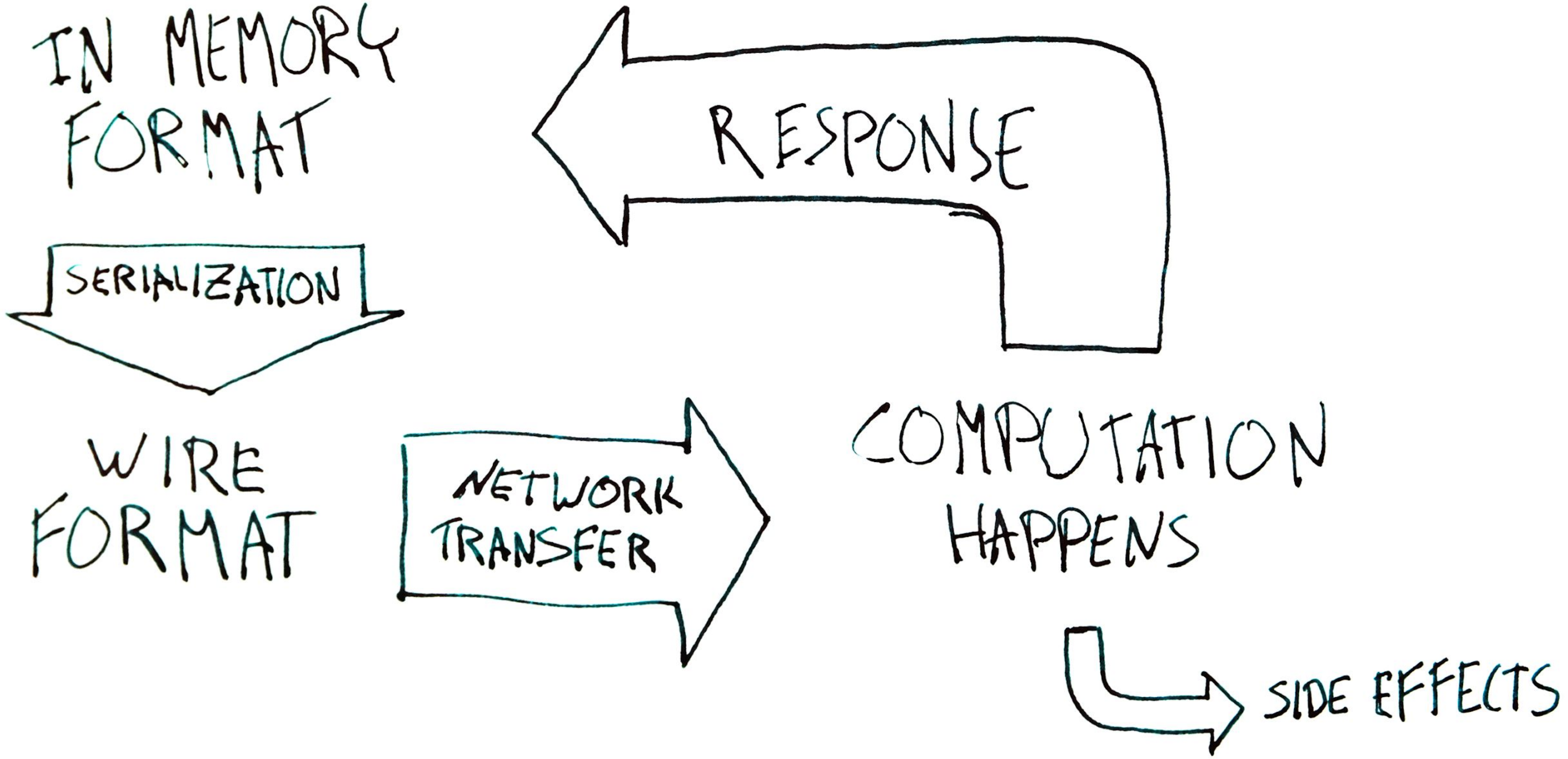
gRPC + gRPC/REST gateway

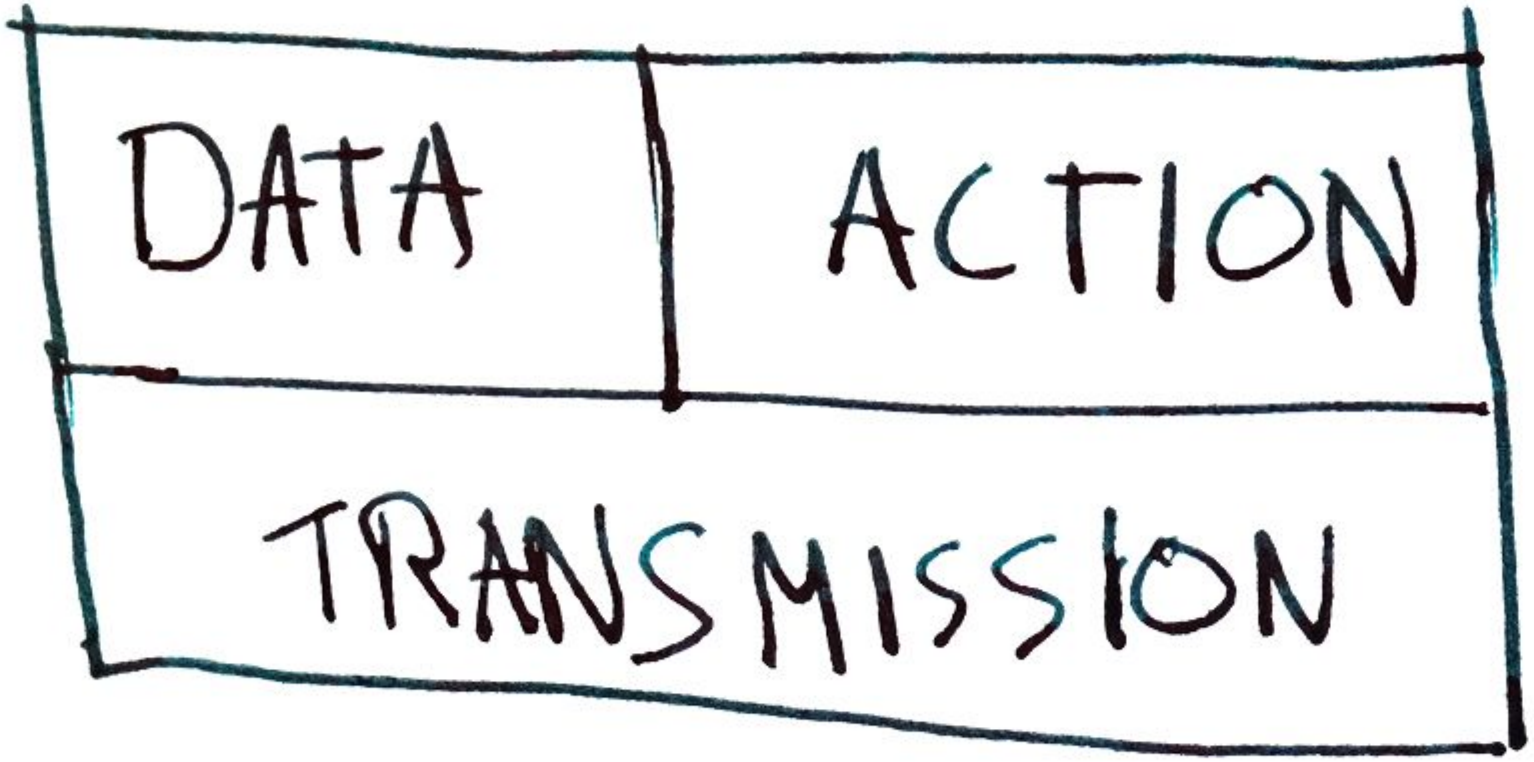
```
message Item {
     string size = 1;
}

message CreateItemRequest {
     string name = 1;
     Item item = 2;
}

message CreateItemResponse {
     int response_code = 1;
}

...
```

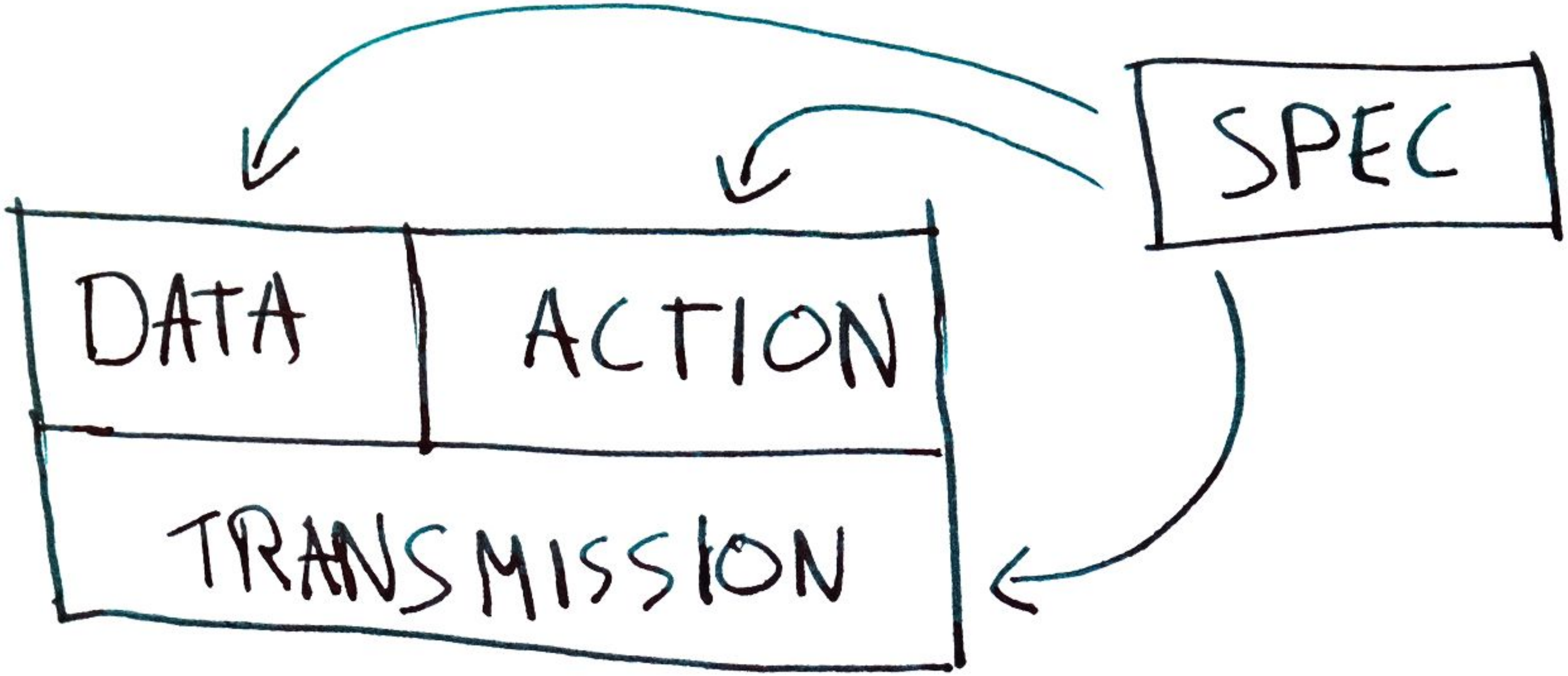
```
service InventoryManagementService {
     rpc Create(CreateItemRequest) returns (CreateItemResponse) {
          option (google.api.http) = {
               post: "/items/{name}"
               body: "item"
          };
     }
     rpc Update(UpdateItemRequest) returns (UpdateItemResponse) {
          option (google.api.http) = {
               put: "/items/{name}"
               body: "item"
          };
     }
     ... // Get, Delete, ...
}
```

That’s great, but I need 4 handlers and 8 data models for every resource.

API systems are opinions about how data should be transmitted between client and server.


IN MEMORY FORMAT
SERIALIZATION
WIRE FORMAT
NETWORK TRANSFER
COMPUTATION HAPPENS
RESPONSE
SIDE EFFECTS


DATA
ACTION
TRANSMISSION


DATA
ACTION
TRANSMISSION
SPEC

The Kubernetes Resource Model is a set of requirements on all aspects of the API call.

$$\begin{tikzcd}
\text{STANDARD DATA STRUCTURE} \arrow[dr] & & \text{STANDARD VERBS} \arrow[dl] \\
& \text{STANDARD RESOURCE} &
\end{tikzcd}$$

The Kubernetes Resource Model goes beyond being object oriented.

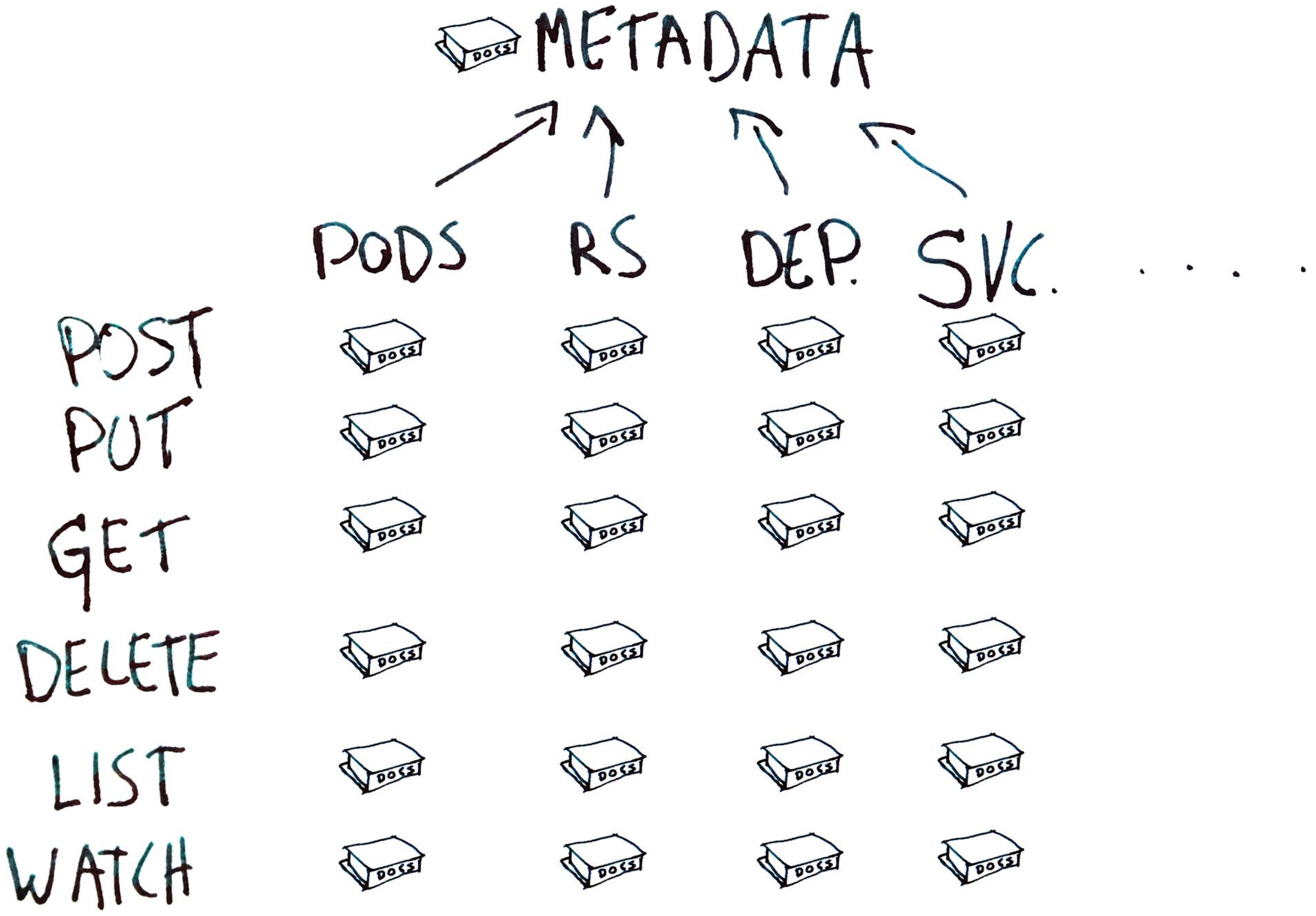
Complexity Management

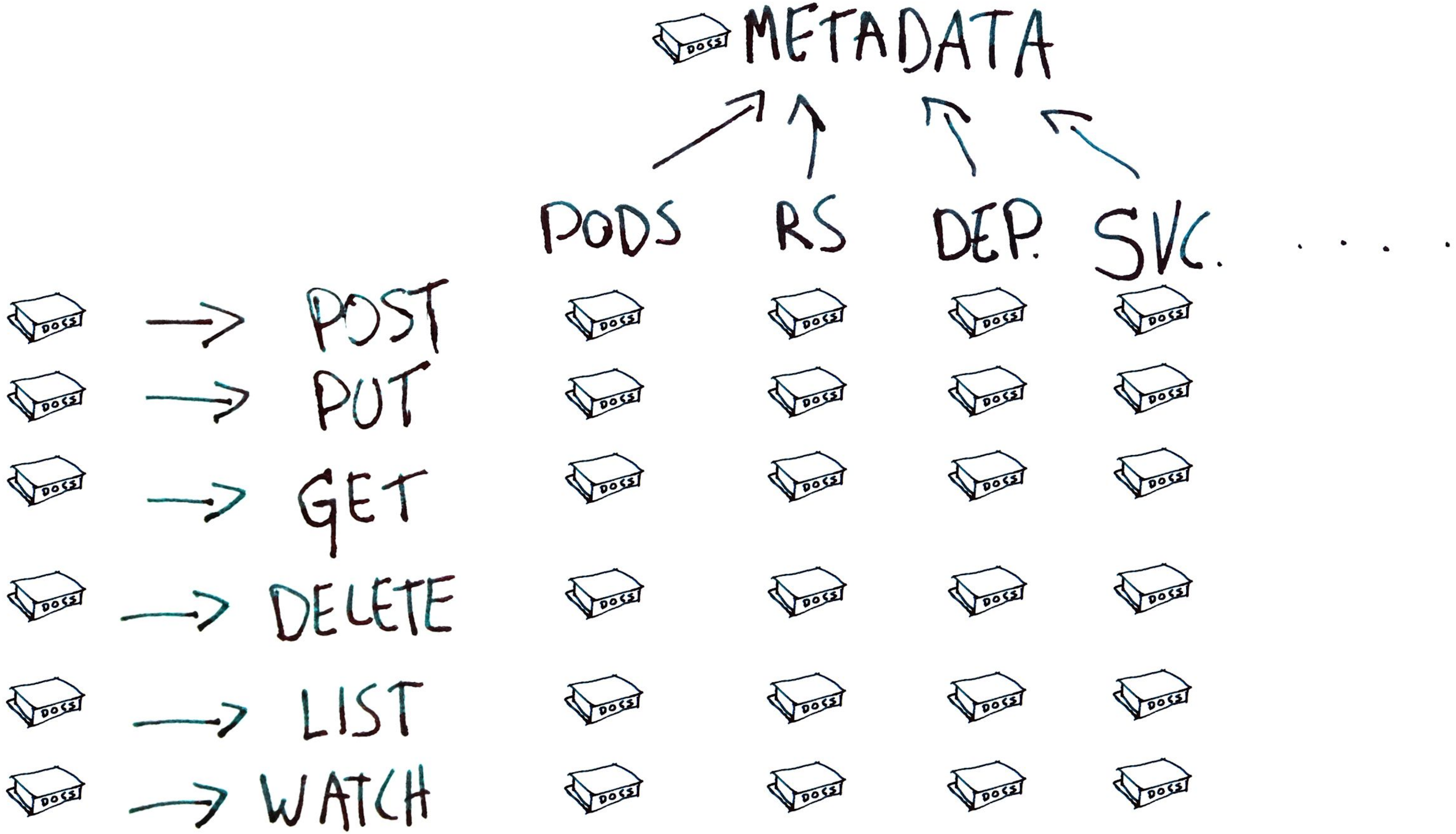
API Operation Complexity

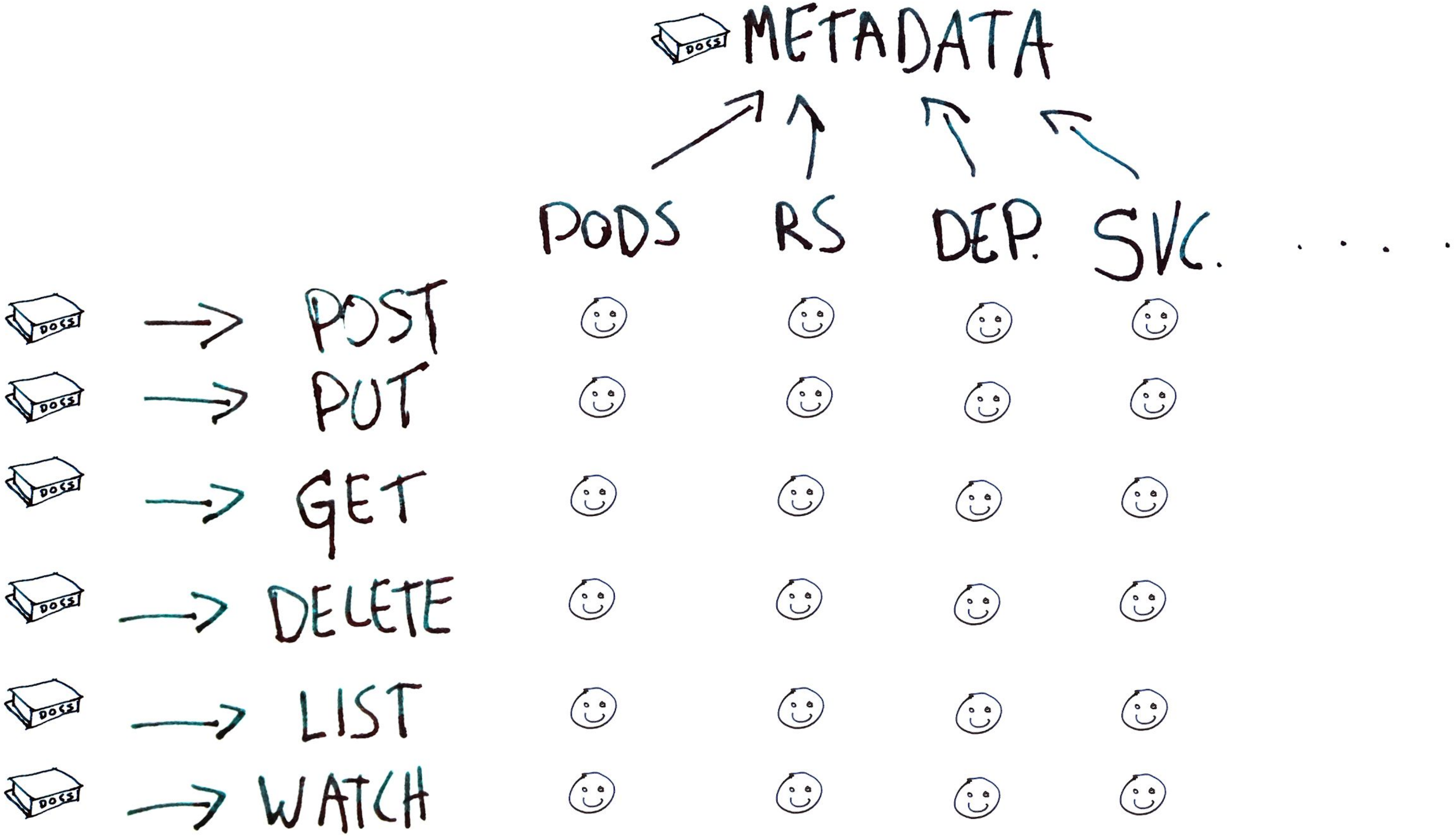
	PODS	RS	DEP.	SVC.	. . . .
POST					
PUT					
GET					
DELETE					
LIST					
WATCH					

	PODS	RS	DEP.	SVC.	. . . .
POST	? ? 😕	? ? 😕	? ? 😕	? ? 😕	
PUT	? ? 😕	? ? 😕	? ? 😕	? ? 😕	
GET	? ? 😕	? ? 😕	? ? 😕	? ? 😕	
DELETE	? ? 😕	? ? 😕	? ? 😕	? ? 😕	
LIST	? ? 😕	? ? 😕	? ? 😕	? ? 😕	
WATCH	? ? 😕	? ? 😕	? ? 😕	? ? 😕	

	PODS	RS	DEP.	SVC.	. . . .
POST	DOCS	DOCS	DOCS	DOCS	
PUT	DOCS	DOCS	DOCS	DOCS	
GET	DOCS	DOCS	DOCS	DOCS	
DELETE	DOCS	DOCS	DOCS	DOCS	
LIST	DOCS	DOCS	DOCS	DOCS	
WATCH	DOCS	DOCS	DOCS	DOCS	


METADATA
DOCS
PODS
RS
DEP.
SVC.
. . . .
POST
PUT
GET
DELETE
LIST
WATCH


DOCS
METADATA
PODS
RS
DEP.
SVC.
POST
PUT
GET
DELETE
LIST
WATCH



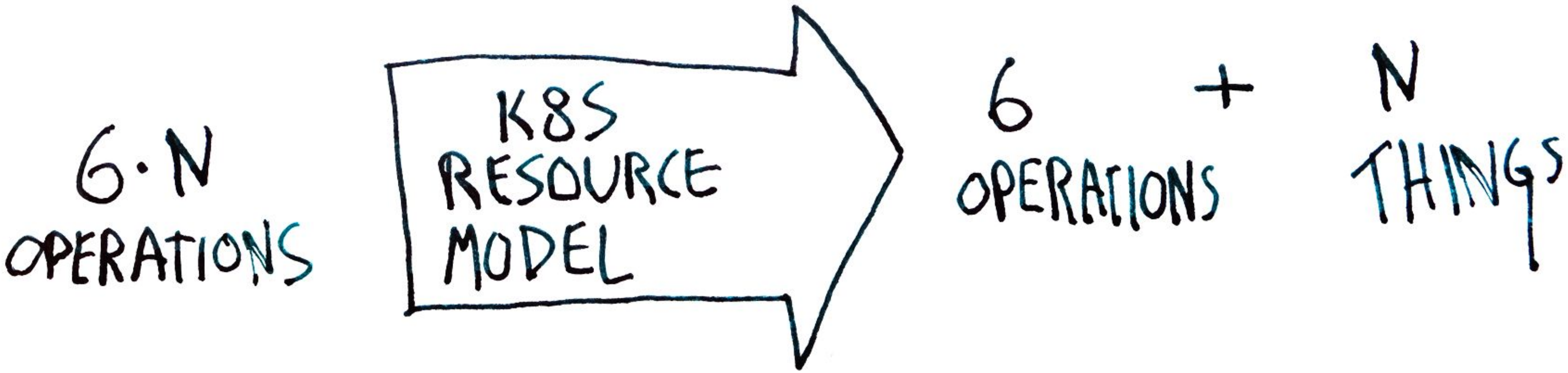
METADATA
DOCS
PODS
RS
DEP.
SVC.
. . . .
POST
PUT
GET
DELETE
LIST
WATCH

POD	NODE	SVC . . .	RBAC . . .	MY CR	YOUR CR	THEIR CR
POST	POST	POST	POST	POST	POST	POST
PUT	PUT	PUT	PUT	PUT	PUT	PUT
GET	GET	GET	GET	GET	GET	GET
DELETE	DELETE	DELETE	DELETE	DELETE	DELETE	DELETE
LIST	LIST	LIST	LIST	LIST	LIST	LIST
WATCH	WATCH	WATCH	WATCH	WATCH	WATCH	WATCH

6·N
OPERATIONS

6·N
OPERATIONS

K8S
RESOURCE
MODEL


6·N OPERATIONS
K8S RESOURCE MODEL
6 OPERATIONS
+
N THINGS

State Complexity

	POD	
NODE	✓ ✓	✓ x
	x ✓	x x

$$\overset{\text{NODE}}{5} \cdot \overset{\text{NODE}}{5} \overset{N}{\cdot\,\cdot\,\cdot} = 5^{N}$$

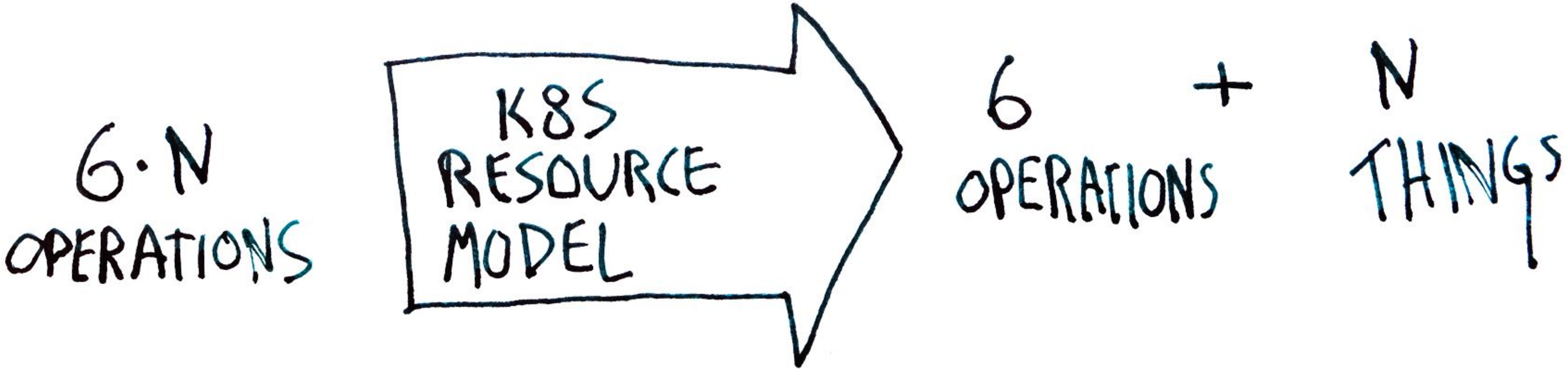
Splitting problems into small pieces which can be acted on concurrently by controllers and users results in flexible, future-proof systems.

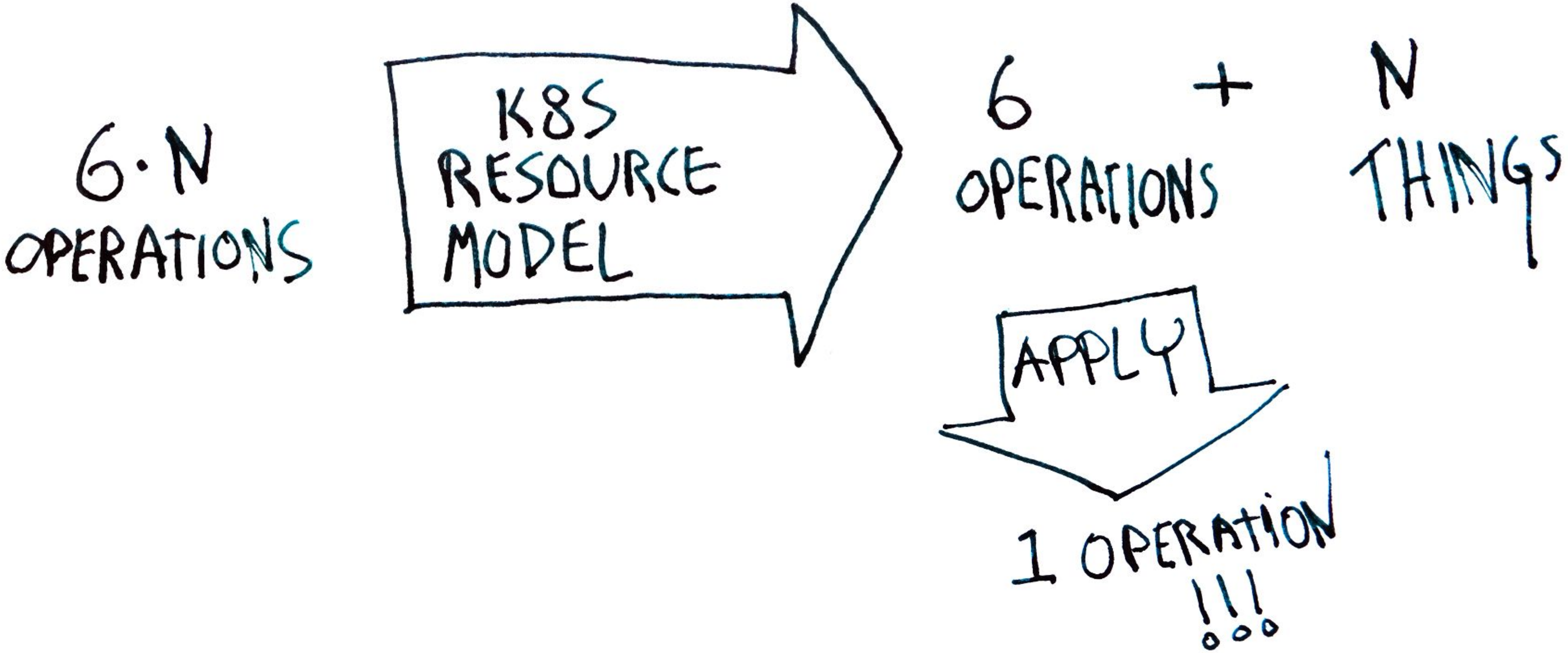
CONTROLLER

$\times \longrightarrow +$

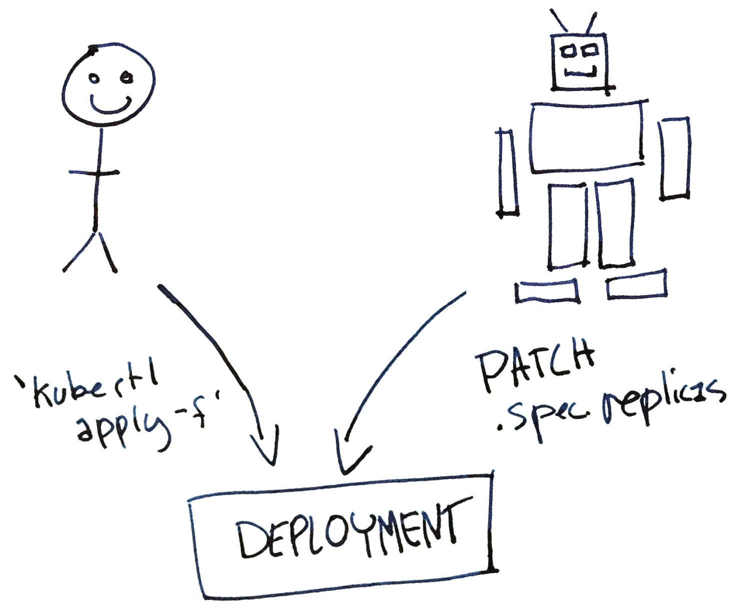
$5^N \longrightarrow N \cdot 5$

“Apply” allows users and systems to cooperatively determine the desired state of an object.


6·N
OPERATIONS
K8S
RESOURCE
MODEL
6 + N
OPERATIONS
THINGS


6·N
OPERATIONS
K8S
RESOURCE
MODEL
6 + N
OPERATIONS
THINGS
APPLY
1 OPERATION
!!!

This opens up new avenues for cooperatively determining the state of a resource.



The Kubernetes Resource Model makes `apply` possible.

Why will you see APIs following the Kubernetes Resource Model in the future?

It’s because a lot of real-word systems are a good fit for this resource model.

Virtual resources:
VMs, load balancers, database instances, service mesh endpoints, ...

Physical resources:
network switches, routers, …

Physical resources:
smart light bulbs, door locks, thermostats?

Can you keep your system’s
entire state diagram in your head
all at once?

What happens to your if statements and flow diagram if you add a few new states?

The Kubernetes Resource Model allows you to effectively manage the complexity of your API ecosystem.

…and that is why you will encounter this style of API in the future.

KubeCon | CloudNativeCon

Europe 2018

Thank you