

**Яндекс**

Universal control plane(s)

Infra sucks!



**Infra sucks!**

Infra sucks!

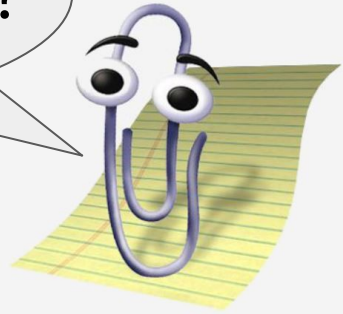
Sure it does, but why?



Infra sucks!

Sure it does, but why?

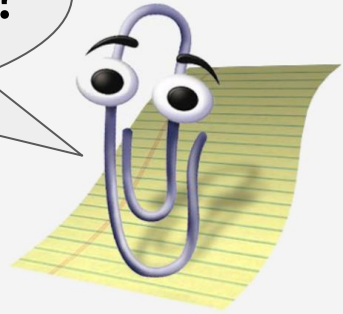
Because...  
I suck at it :/



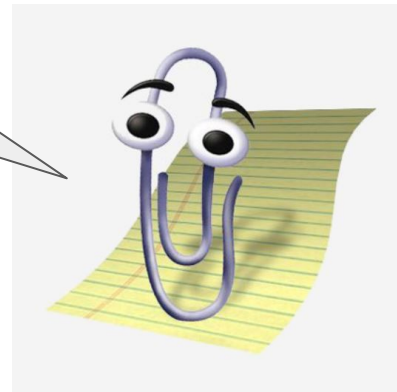
Infra sucks!

Sure it does, but why?

**Because...  
I suck at it :/**



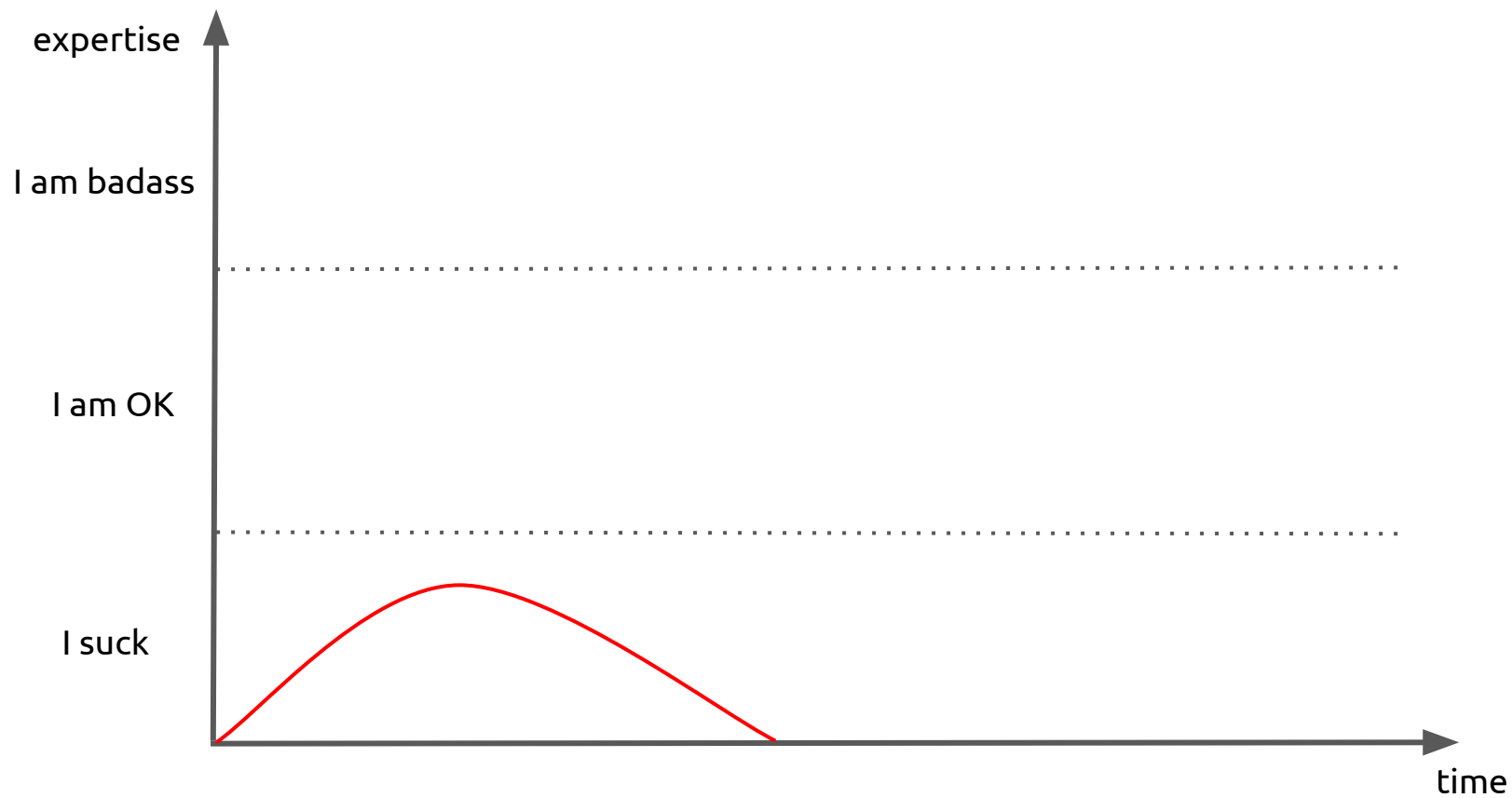
Let's fix this!

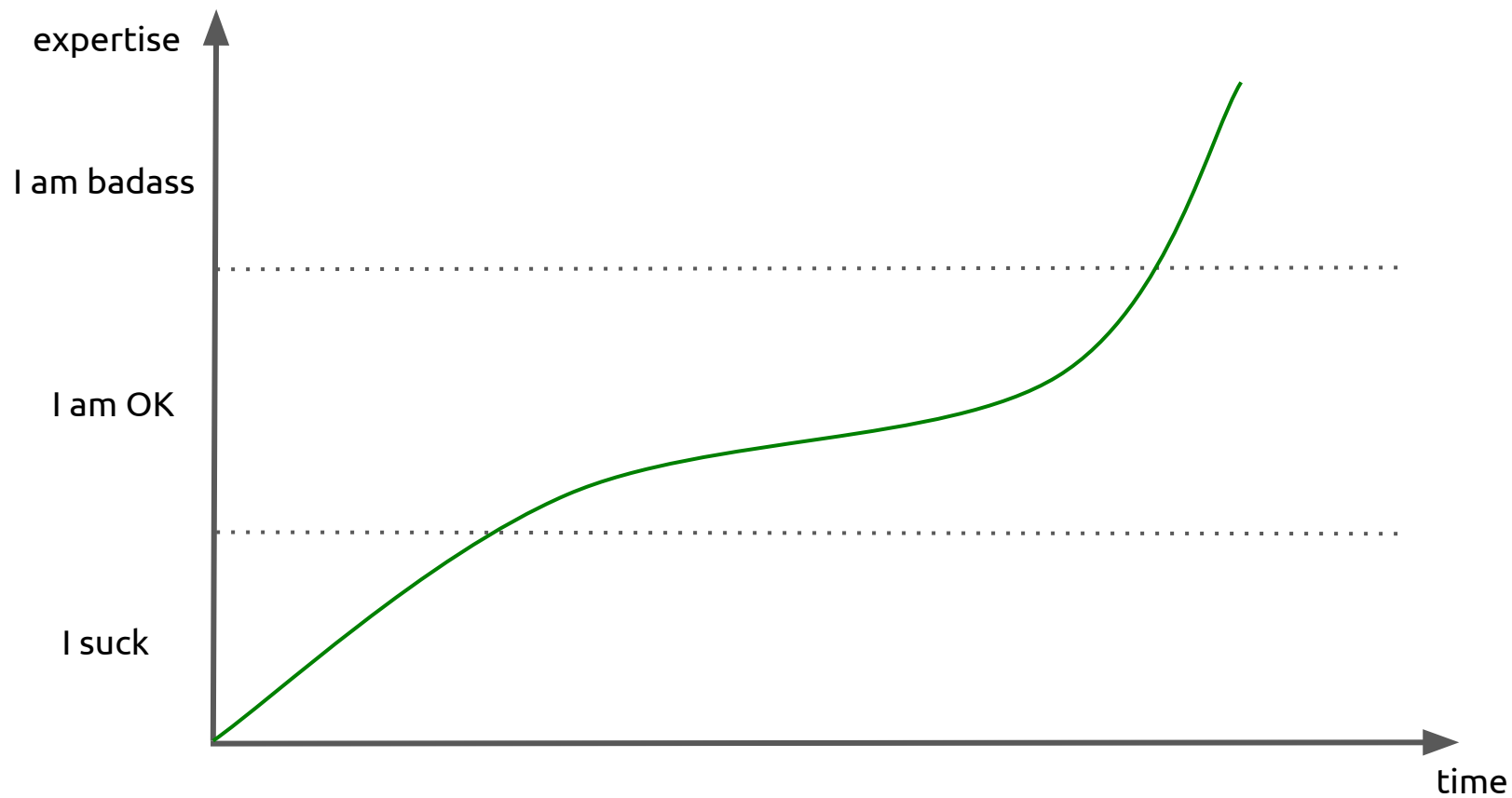


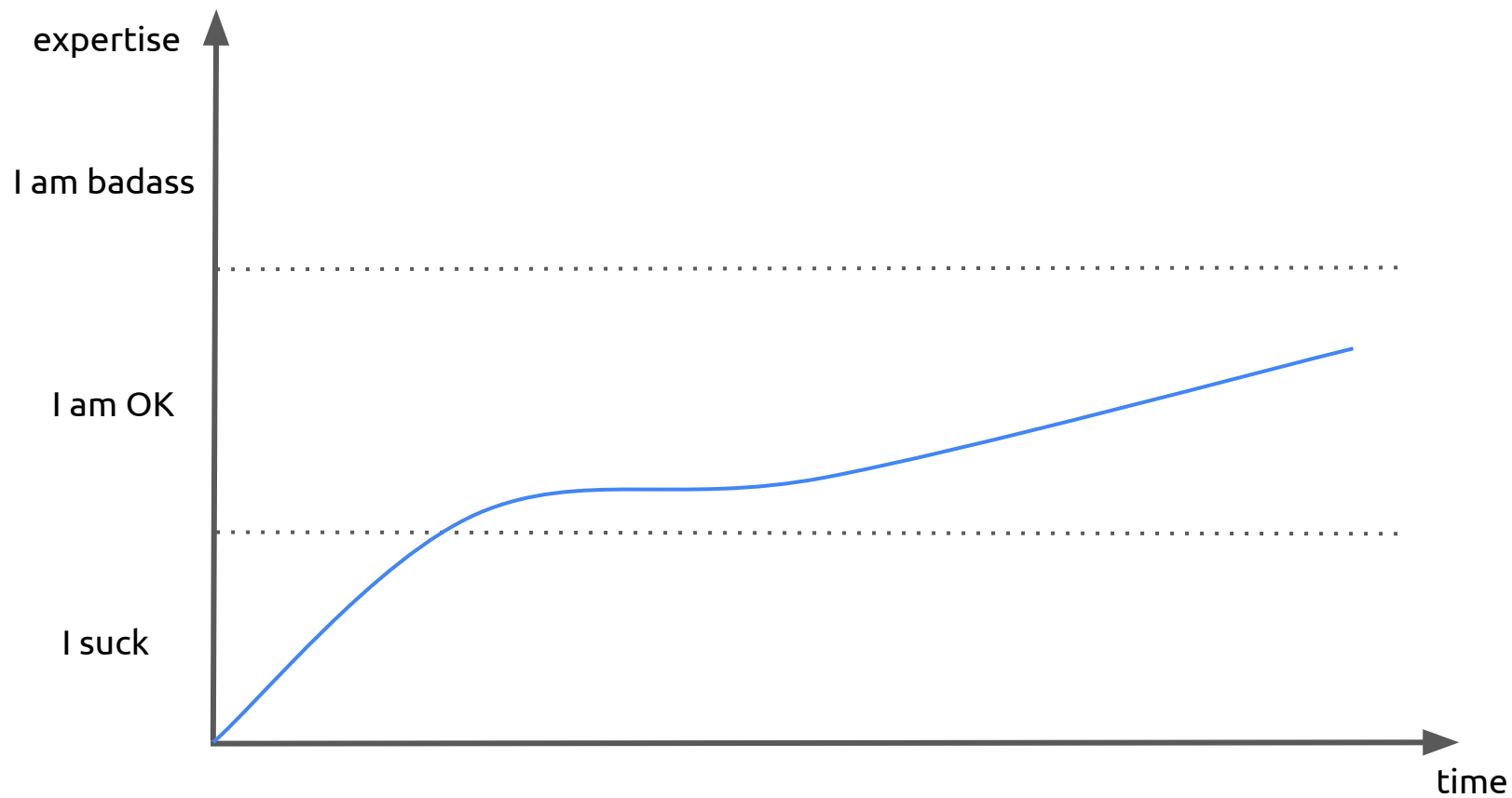


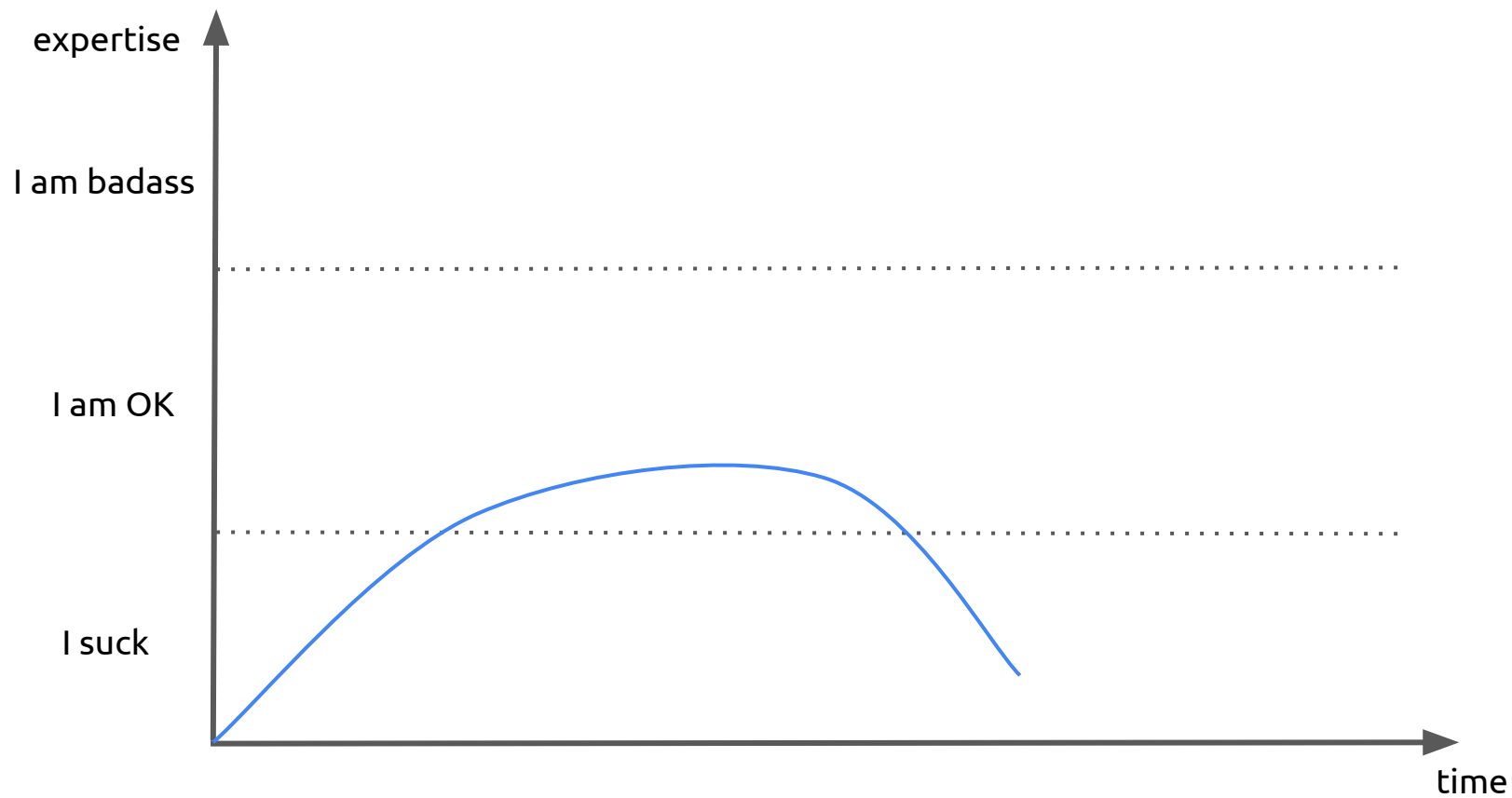


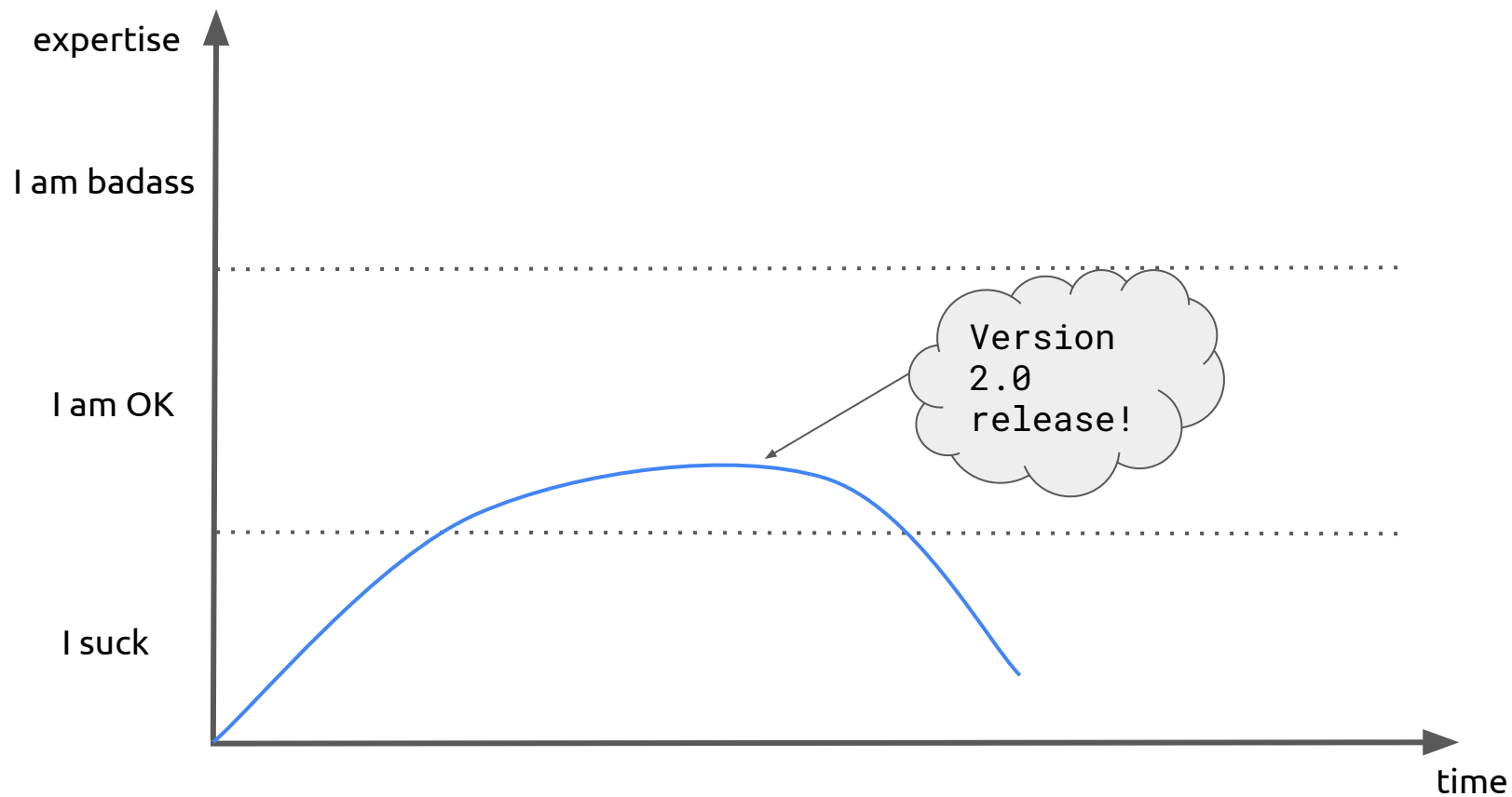


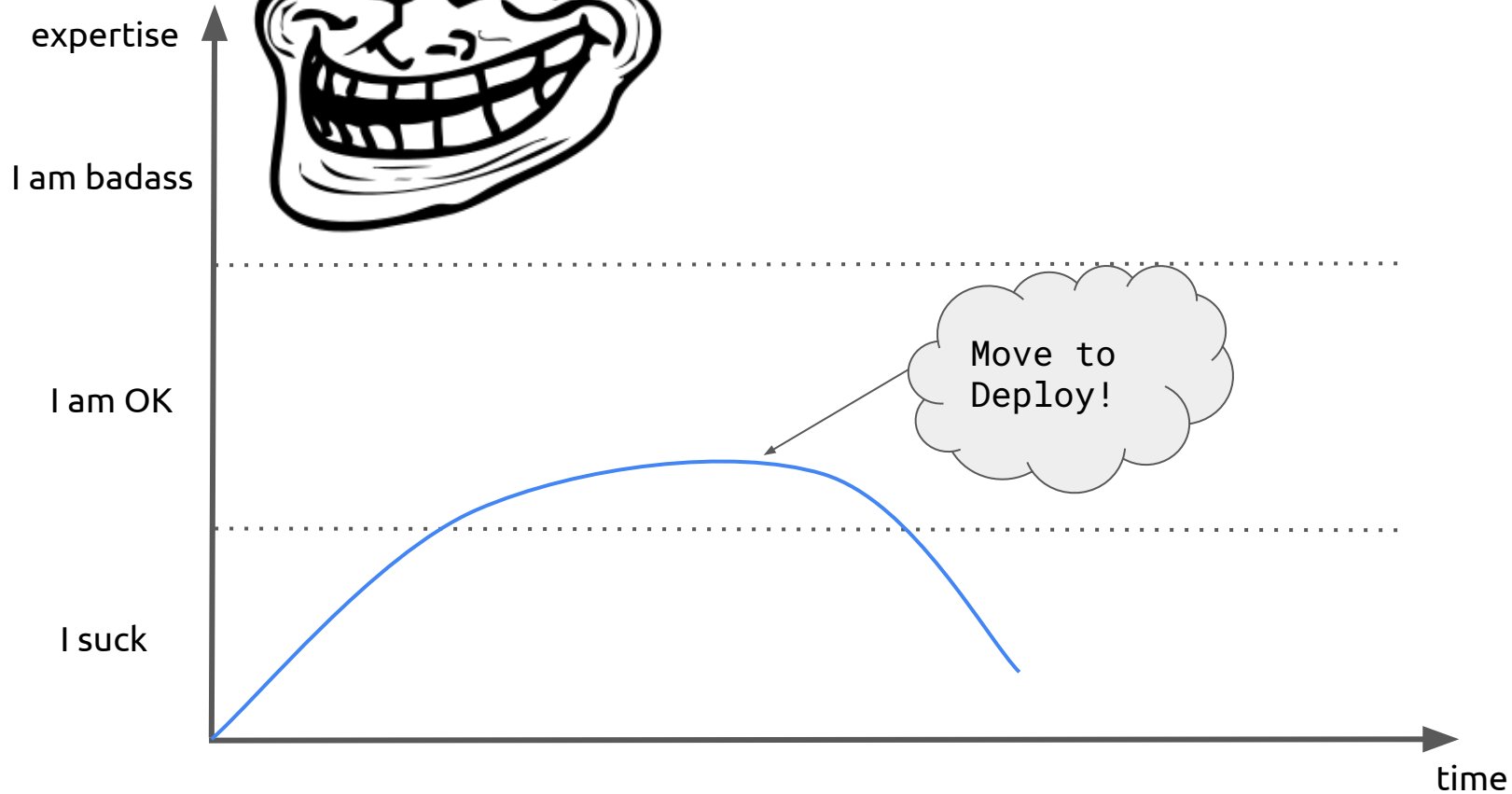




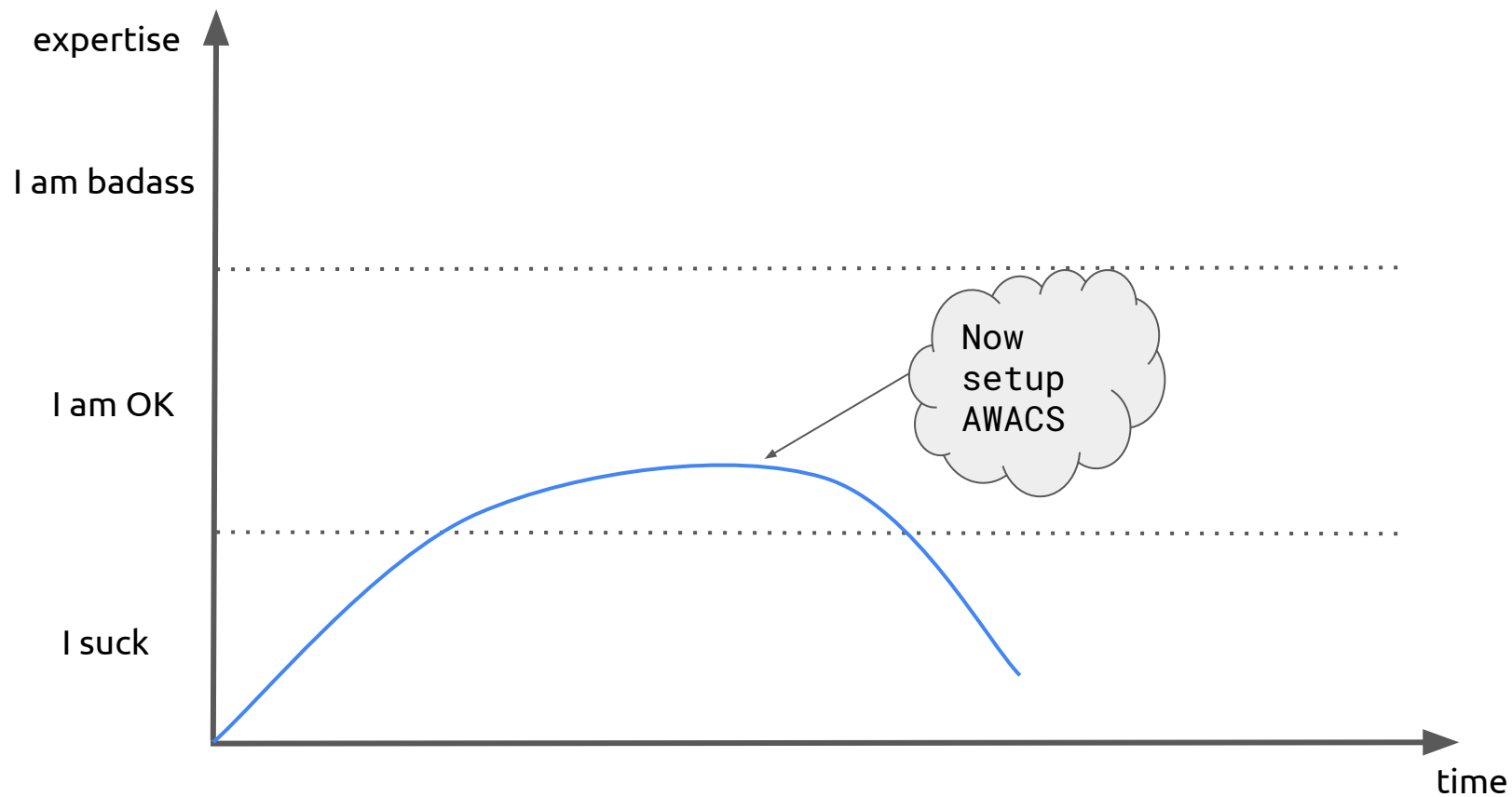


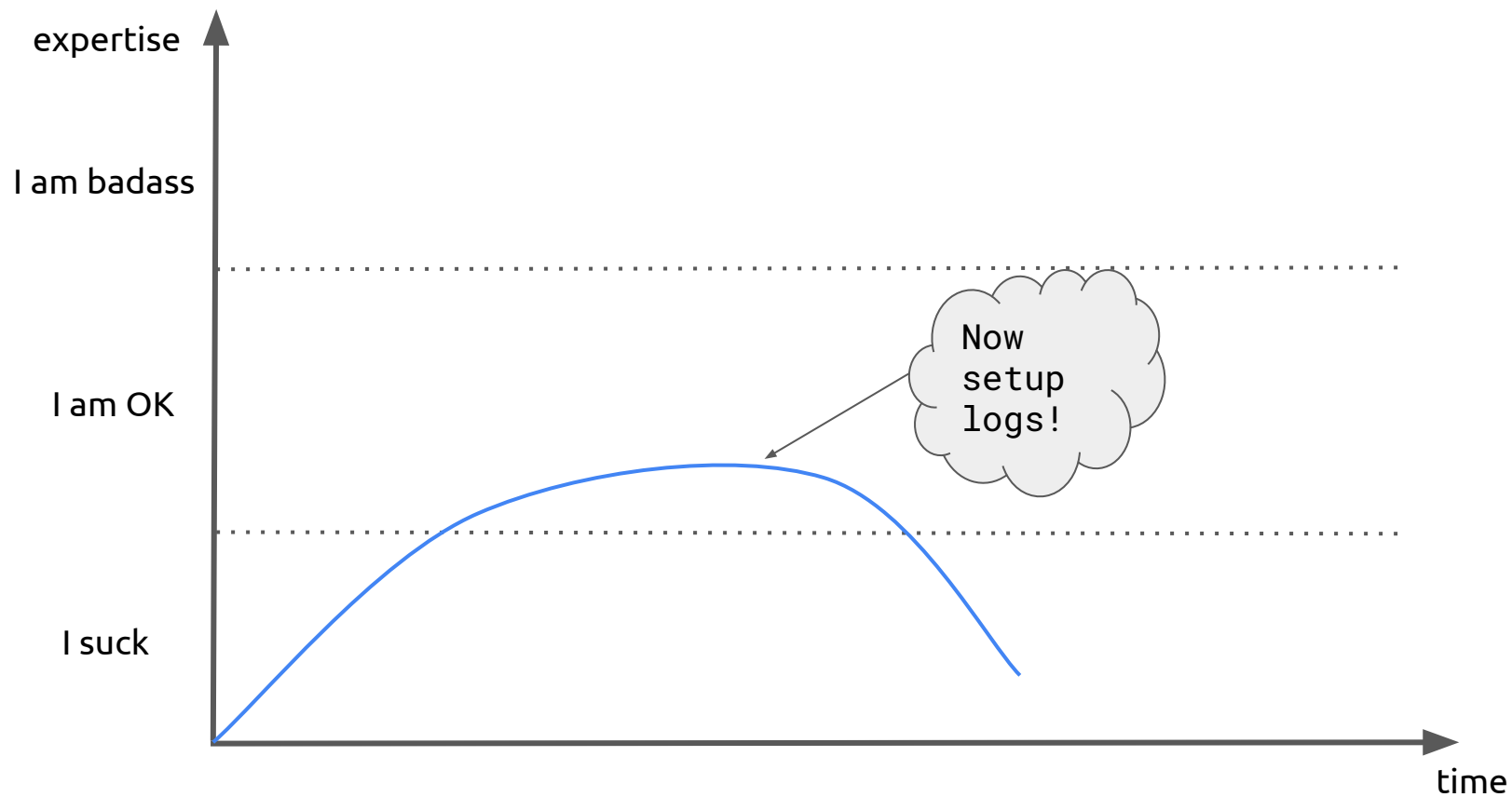












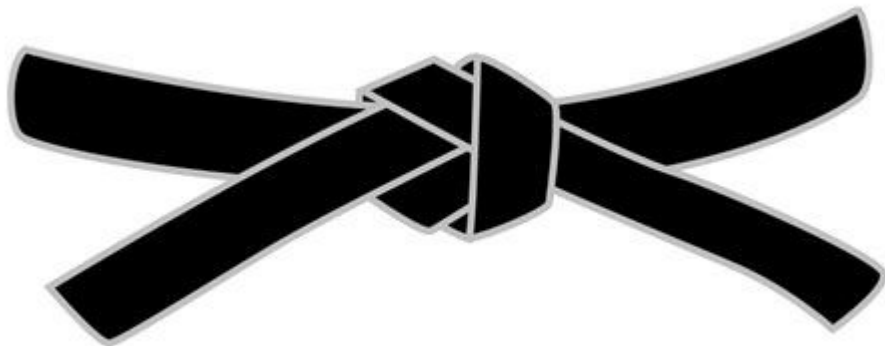
Ещё проблемы:  
namespace в juggler/awacs  
roles/admins/downtimers

Ещё проблемы для провайдеров:  
аутентификация, авторизация

Okay, experts. In what?

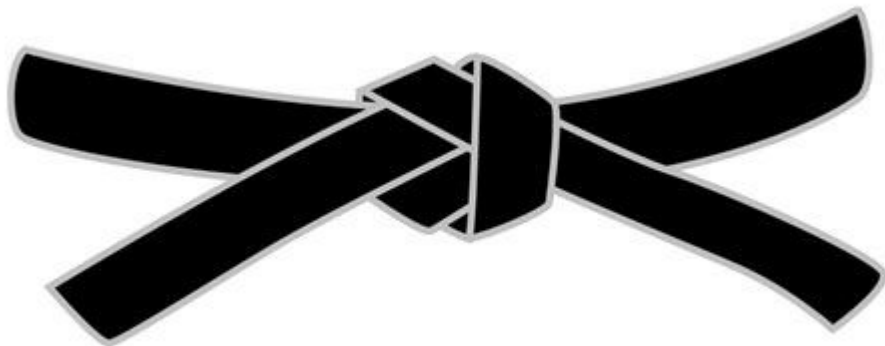
# Tools to master:

- Juggler (jctl, UI, ansible, API)
- Deploy (dctl, UI, API)
- yasm (custom generators, UI, Jinja)
- logbroker
- Nanny
- AWACS (UI, awacsctl2)
- MDB (UI, API?)



# Tools to master:

- ~~Juggler (jctl, UI, ansible, API)~~
- ~~Deploy (dctl, UI, API)~~
- ~~yasm (custom generators, UI, Jinja)~~
- ~~logbroker~~
- ~~Nanny~~
- ~~AWACS~~



# Areas to master

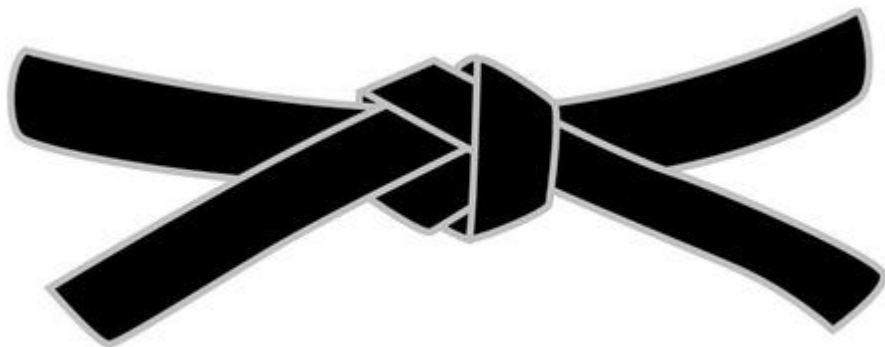
- Software engineering
- DevOps
- Systems design
- Distributed systems





# Tools to master:

- infractl



Far far in the future...

<demo placeholder>

Back to present time...

# Table of contents

- Declarative configuration management
- Kubernetes resource model
- Current state

# Declarative configuration management

- do not specify **actions**
- specify desired outcome

```
$ cat frontend.yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  # modify replicas according to your case
  replicas: 3
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
        - name: kino-app
          image: registry.yandex.ru/kino/kino-frontend:v3
```

```
$ kubectl apply -f frontend.yaml
```



```
$ kubectl apply -f frontend.yaml
```

```
$ kubectl get rs
```

NAME	DESIRED	CURRENT	READY	AGE
frontend	3	3	3	6s

```
$ kubectl apply -f frontend.yaml
```

```
$ kubectl get rs
```

NAME	DESIRED	CURRENT	READY	AGE
frontend	3	3	3	6s

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
frontend-b2zdvd	1/1	Running	0	6m36s
frontend-vcmts	1/1	Running	0	6m36s
frontend-wtsmm	1/1	Running	0	6m36s







ONE RING TO RULE  
THEM ALL



Azure



Yandex Cloud

```
$ cat database.yaml
apiVersion: sql.cnrm.cloud.google.com/v1beta1
kind: SQLInstance
metadata:
  name: my-sql-instance
  labels:
    cost-center: "cc9"
spec:
  databaseVersion: MYSQL_5_7
  region: us-central1
  settings:
    tier: db-f1-micro
```

```
-----
apiVersion: sql.cnrm.cloud.google.com/v1beta1
kind: SQLDatabase
metadata:
  name: my-sql-db
  labels:
    cost-center: "cc9"
spec:
  instanceRef:
    name: my-sql-instance
```

```
$ cat database.yaml
apiVersion: sql.cnrm.cloud.google.com/v1beta1
kind: SQLInstance
metadata:
  name: my-sql-instance
  labels:
    cost-center: "cc9"
spec:
  databaseVersion: MYSQL_5_7
  region: us-central1
  settings:
    tier: db-f1-micro
```

```
-----
apiVersion: sql.cnrm.cloud.google.com/v1beta1
kind: SQLDatabase
metadata:
  name: my-sql-db
  labels:
    cost-center: "cc9"
spec:
  instanceRef:
    name: my-sql-instance
```



```
$ cat database.yaml
apiVersion: sql.cnrm.cloud.google.com/v1beta1
kind: SQLInstance
metadata:
  name: my-sql-instance
  labels:
    cost-center: "cc9"
spec:
  databaseVersion: MYSQL_5_7
  region: us-central1
  settings:
    tier: db-f1-micro
```

```
-----
apiVersion: sql.cnrm.cloud.google.com/v1beta1
kind: SQLDatabase
metadata:
  name: my-sql-db
  labels:
    cost-center: "cc9"
spec:
  instanceRef:
    name: my-sql-instance
```

```
apiVersion: database.example.org/v1alpha1
kind: PostgreSQLInstance
metadata:
  name: my-db
  namespace: default
spec:
  parameters:
    storageGB: 20
  compositionSelector:
    matchLabels:
      provider: azure
  writeConnectionSecretToRef:
    name: db-conn
```

Backup slides

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: StorageBucketAllowedLocations
metadata:
  name: allowmultiregions
spec:
  match:
    kinds:
      - apiGroups: ["storage.cnrm.cloud.google.com"]
        kinds: ["StorageBucket"]
  parameters:
    locations:
      - "ASIA"
      - "EU"
      - "US"
```