Chaos Engineering in Action: Enhancing Resilience in Strimzi

Maros Orsak

Senior Software Quality Engineer Redhat

Immortal Dota 2

Henrich Zrncik

Software Quality Engineer RedHat

Warlock lvl 70

Quiz

Content

Quiz

1. **Chaos Engineering**
2. **Target Systems**
3. **Designing Chaos** (Experiments)
4. **Demo (**Simplified)
5. **Conclusion**

System's resilience

1 **Application**

System's resilience

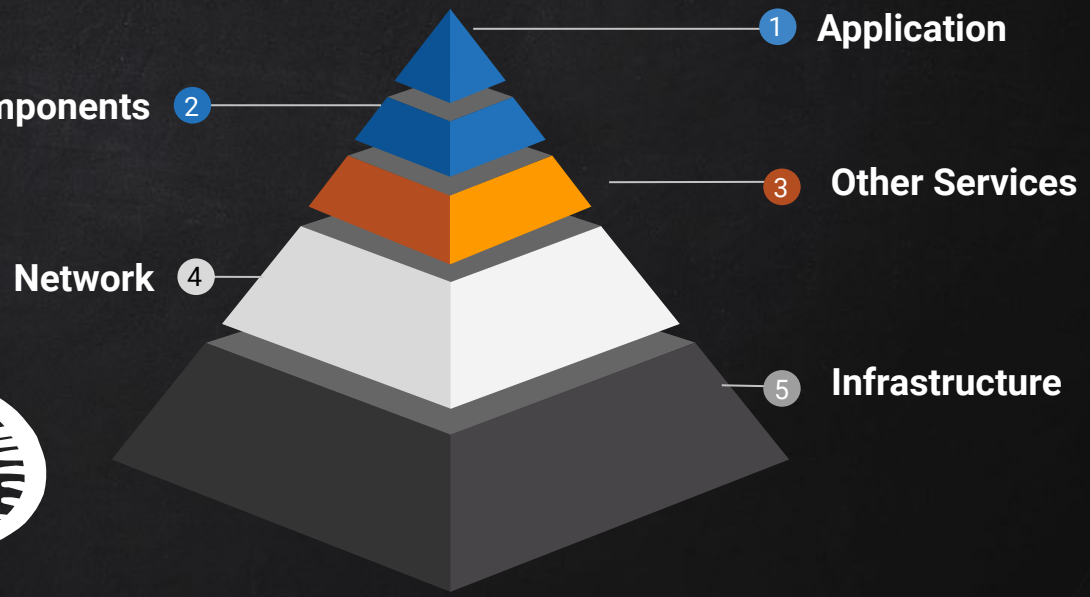
System's resilience

Fallacies of Distributed Systems - L. Peter Deutsch

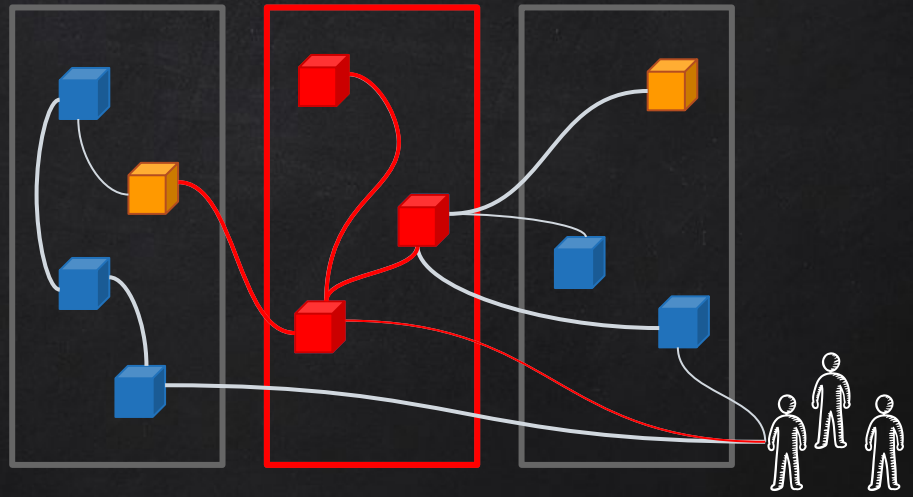
- The network is reliable.
- Latency is zero.
- Bandwidth is infinite.
- The network is secure.

…

"Nonsense...And what's more, it doesn't rhyme. All decent predictions rhyme."
- G. of R.



Chaos Origin

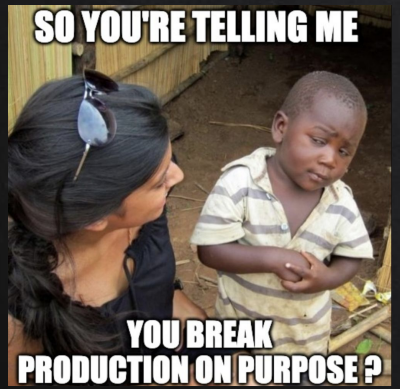


Definition

Experimenting *on a system in order to build* ***confidence*** *in the system's capability to withstand turbulent conditions in* ***production****.*

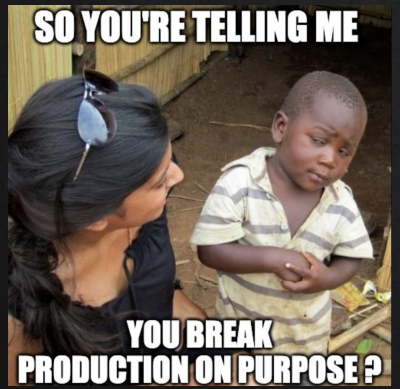
Definition

Experimenting *on a system in order to build* ***confidence*** *in the system's capability to withstand turbulent conditions in* ***production****.*



Definition

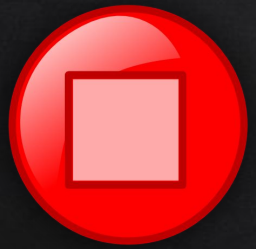
Experimenting *on a system in order to build* ***confidence*** *in the system's capability to withstand turbulent conditions in* ***production****.*



What is the benefit ?

Principles

Minimal blast radius



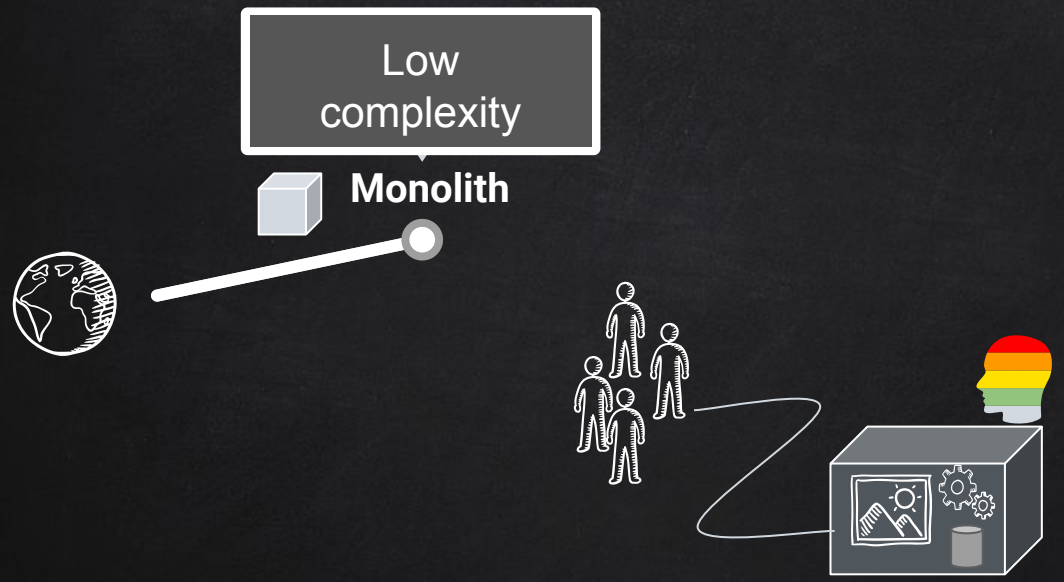
Principles

- Minimal blast radius
- Run in production
- Hypothesis around steady state
- Wary real world's events

Principles

Minimal blast radius → Run in production → Hypothesis around steady state → Wary real world's events → Automatized Continuous run

Architectural shift (timeline)



Architectural shift (timeline)

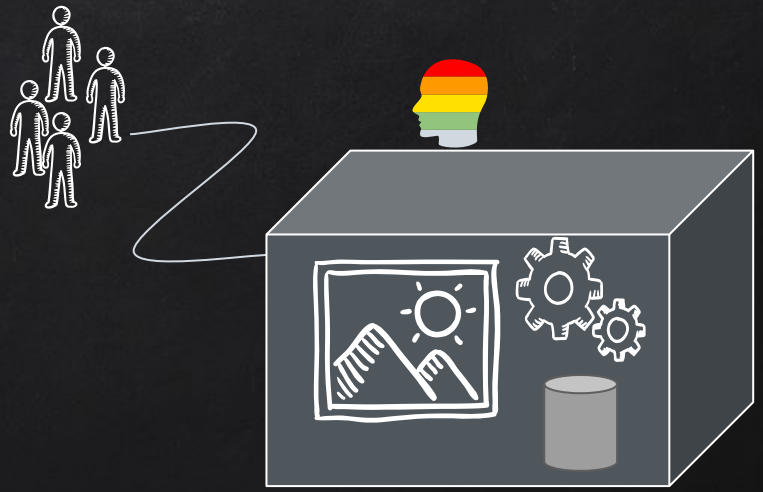
Hard to scale horizontally

Monolith

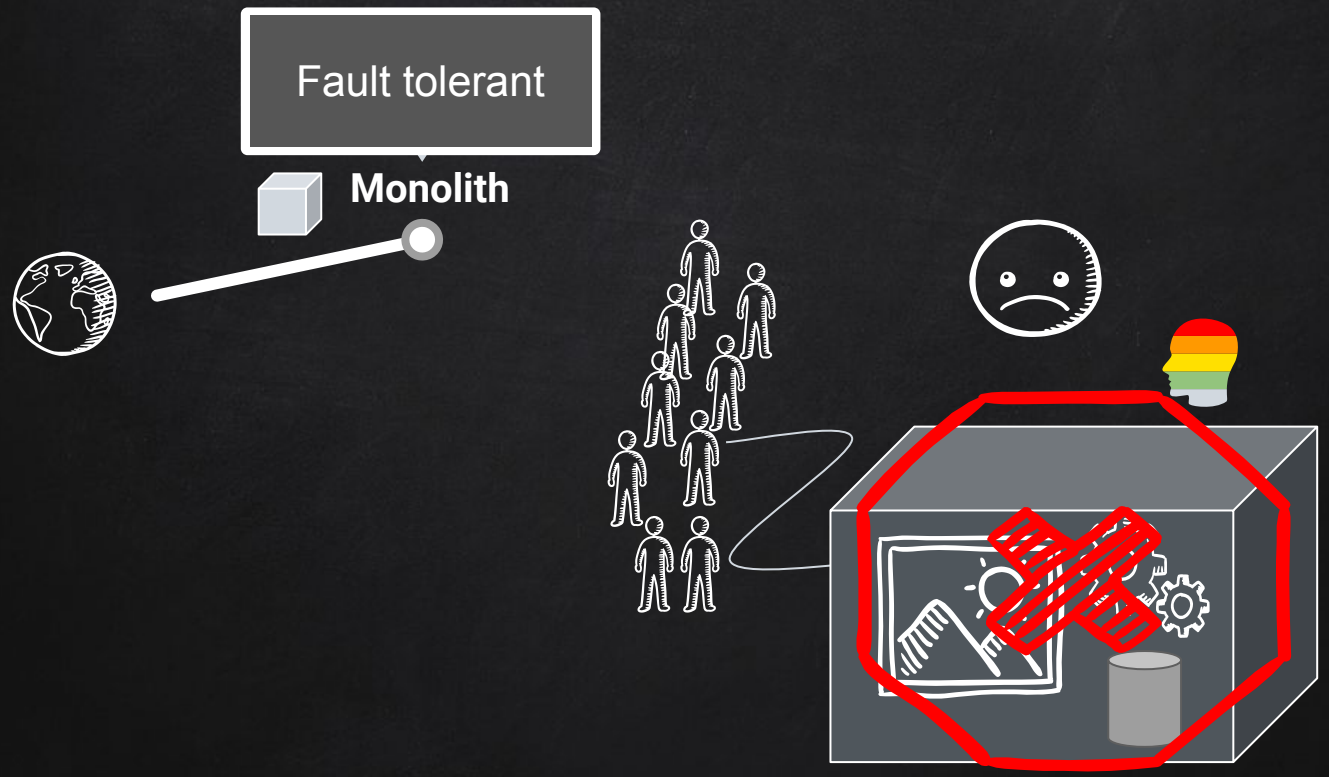
Architectural shift (timeline)

Hard to scale horizontally

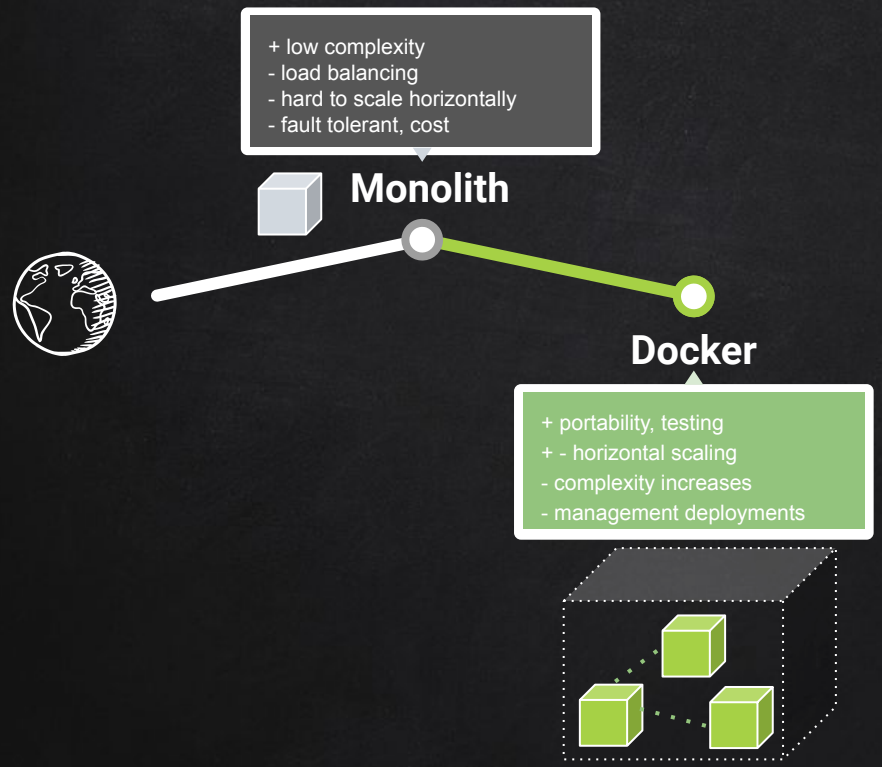
Monolith



Architectural shift (timeline)



Architectural shift (timeline)



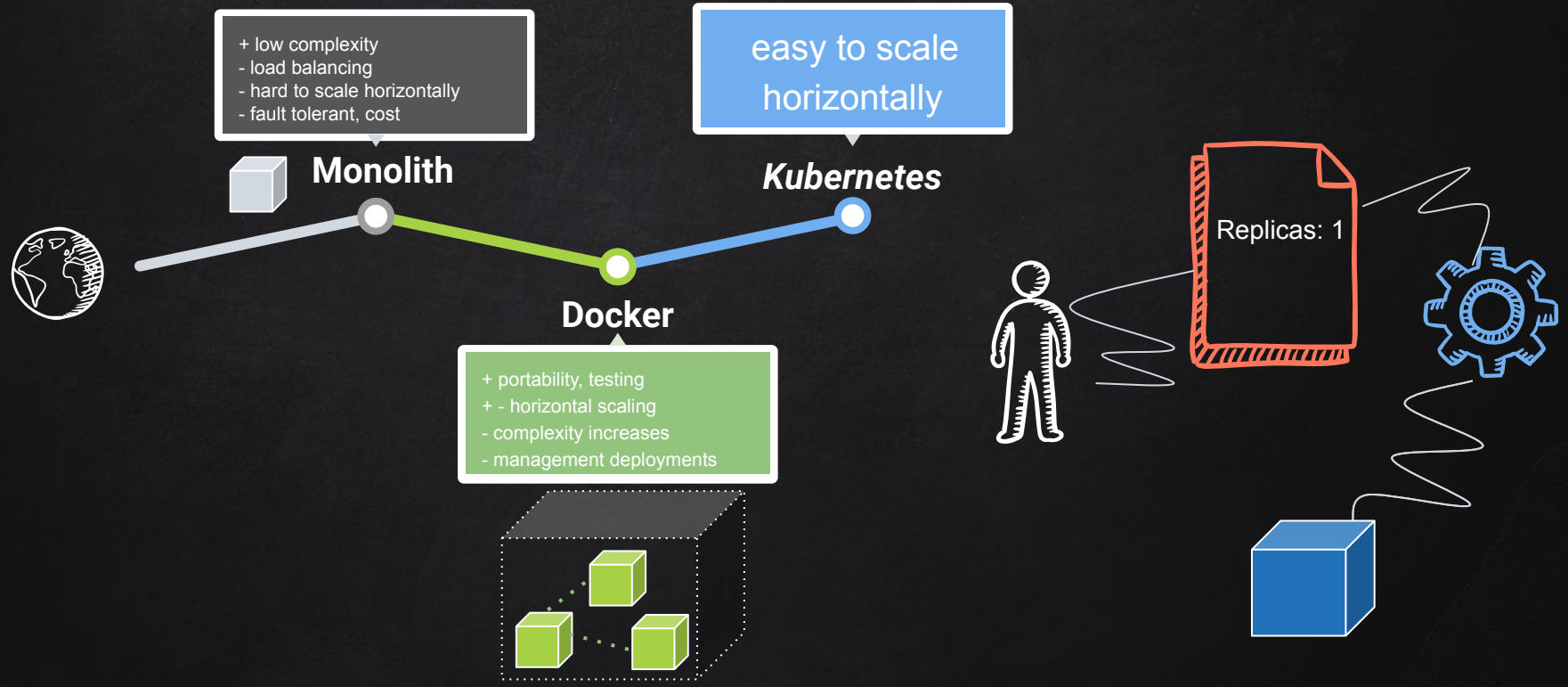
Isolation

Complexity increases

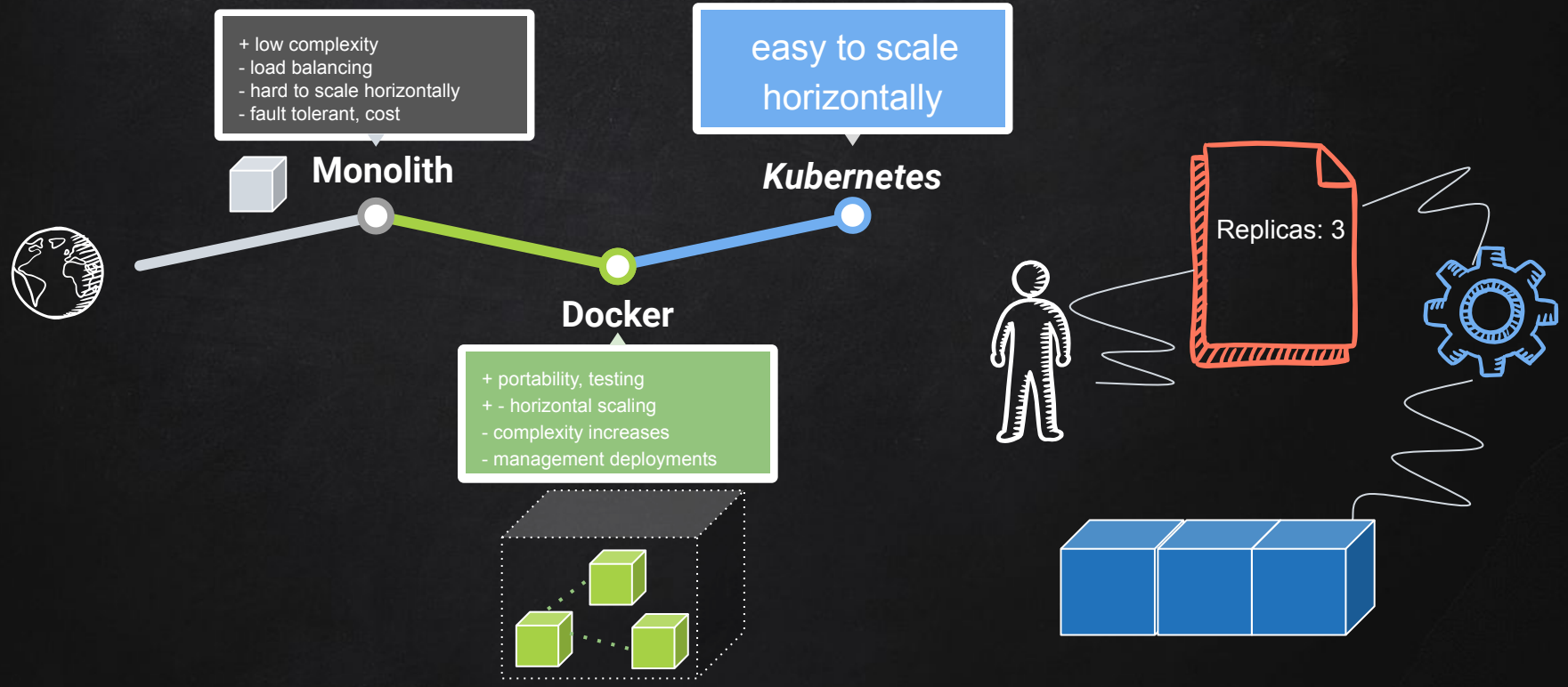
Portability

Horizontal scaling

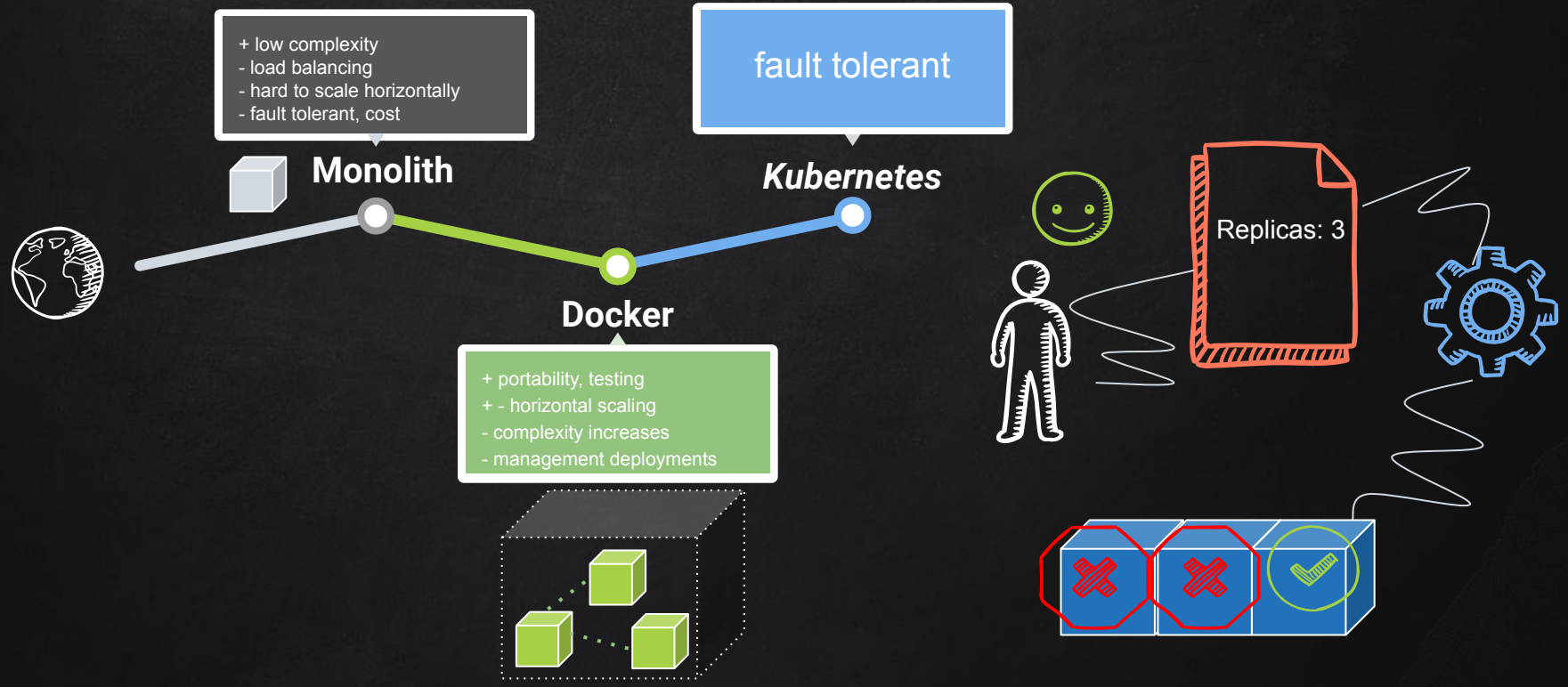
Architectural shift (timeline)



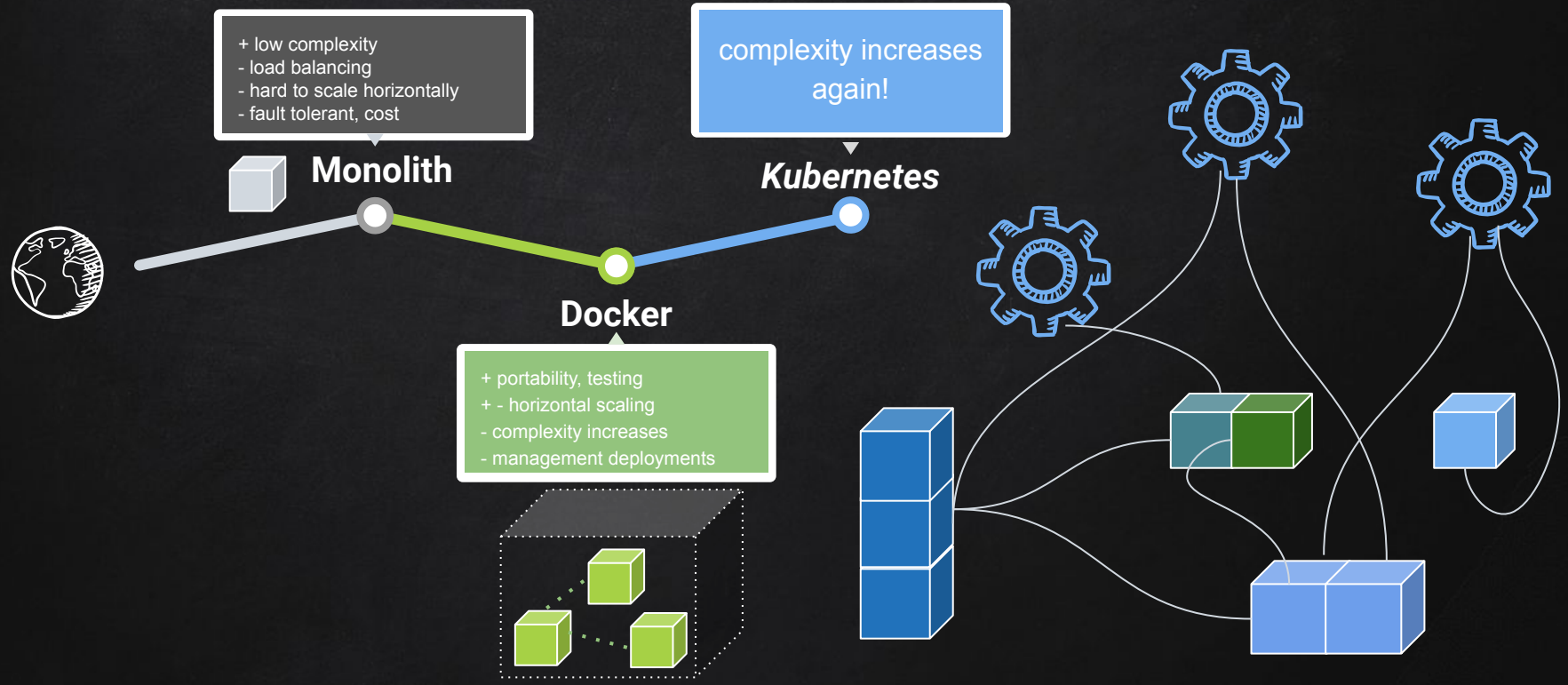
Architectural shift (timeline)



Architectural shift (timeline)



Architectural shift (timeline)



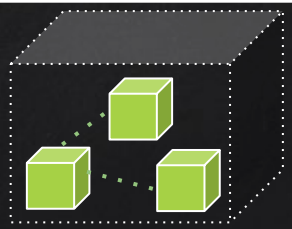
Architectural shift (timeline)

Monolith

+ low complexity
- load balancing
- hard to scale horizontally
- fault tolerant, cost

Docker

+ portability, testing
+ - horizontal scaling
- complexity increases
- management deployments



Kubernetes

+ easy to scale horizontally
+ management of resources
+ fault tolerant
- complexity increases again!

Operators

No single person can grasp the entire system!

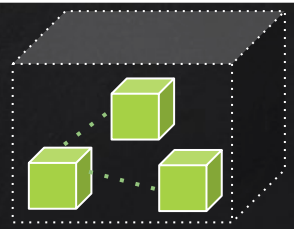
Architectural shift (timeline)

Monolith

+ low complexity
- load balancing
- hard to scale horizontally
- fault tolerant, cost

Docker

+ portability, testing
+ - horizontal scaling
- complexity increases
- management deployments



Kubernetes

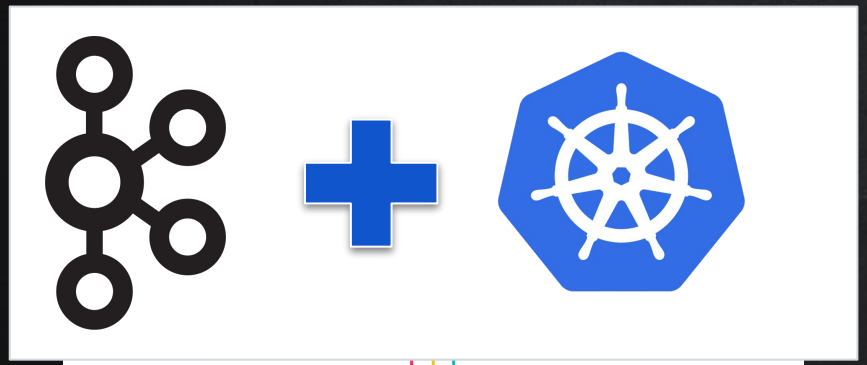
+ easy to scale horizontally
+ management of resources
+ fault tolerant
- complexity increases again!

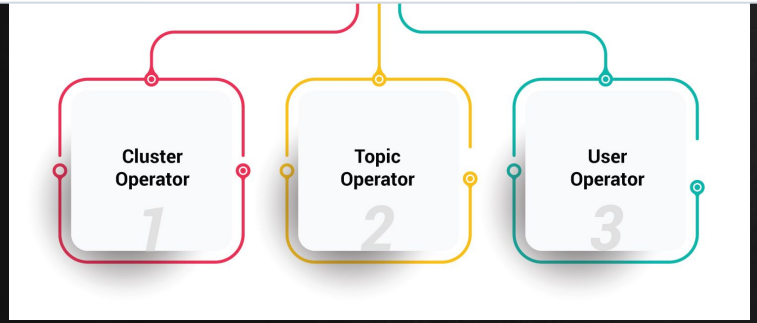
Operators

No single person can grasp the entire system!

where one of such operators is…

Strimzi





Simplifies upgrades of Kafka clusters

Horizontal Scaling

Tracing

Dynamic configuration

Grafana dashboards

Security

Cloud Native Computing Foundation

Cluster Operator

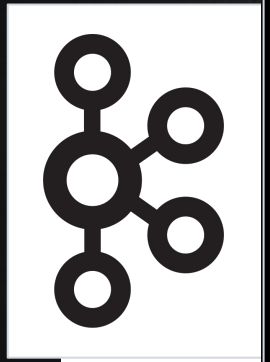
1

al

fana
hboards

uting

Too much unknowns…right…let’s break this down…



l

fana
nboards

uting

Apache Kafka

distributed event streaming platform

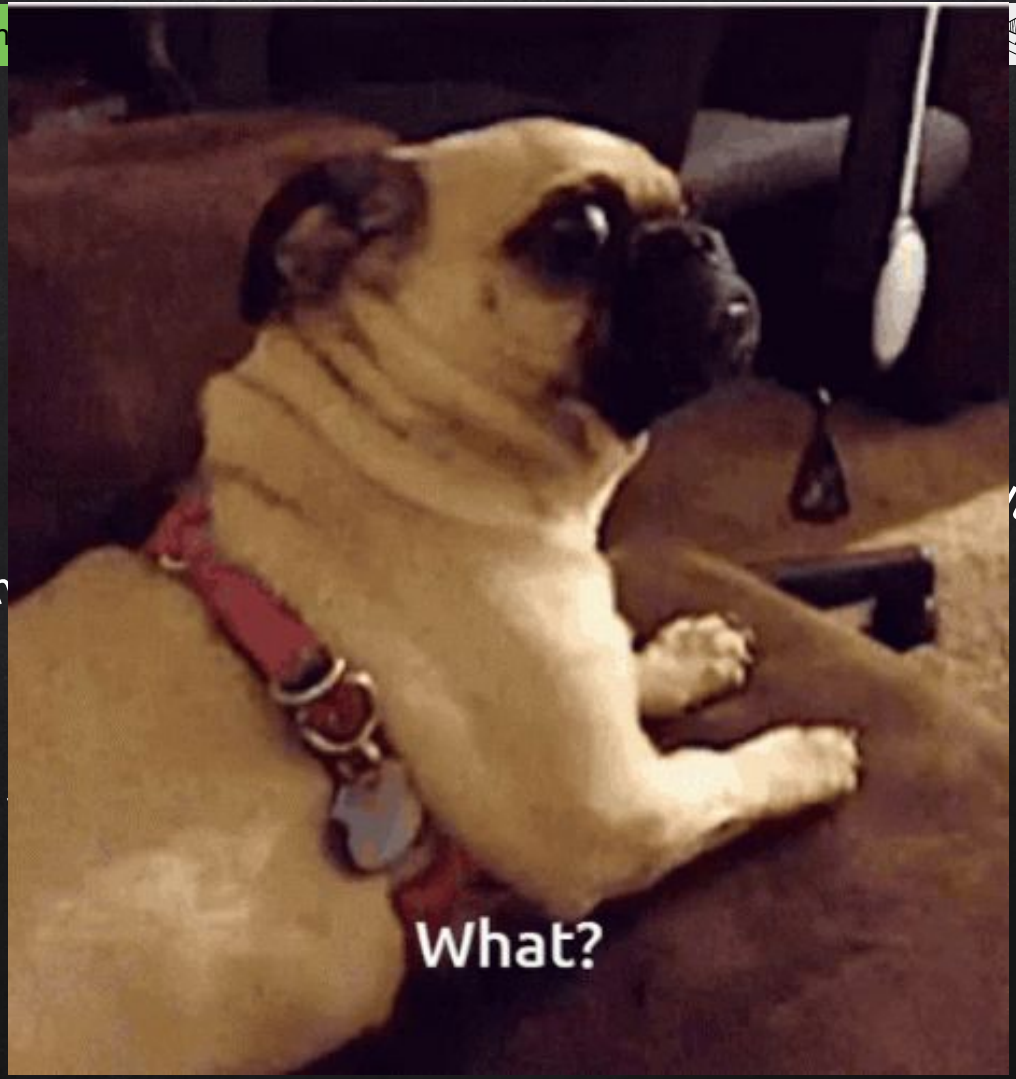
Commit log service

Publish subscribe model

Messaging system

Fault tolerant

distributed even



it log service

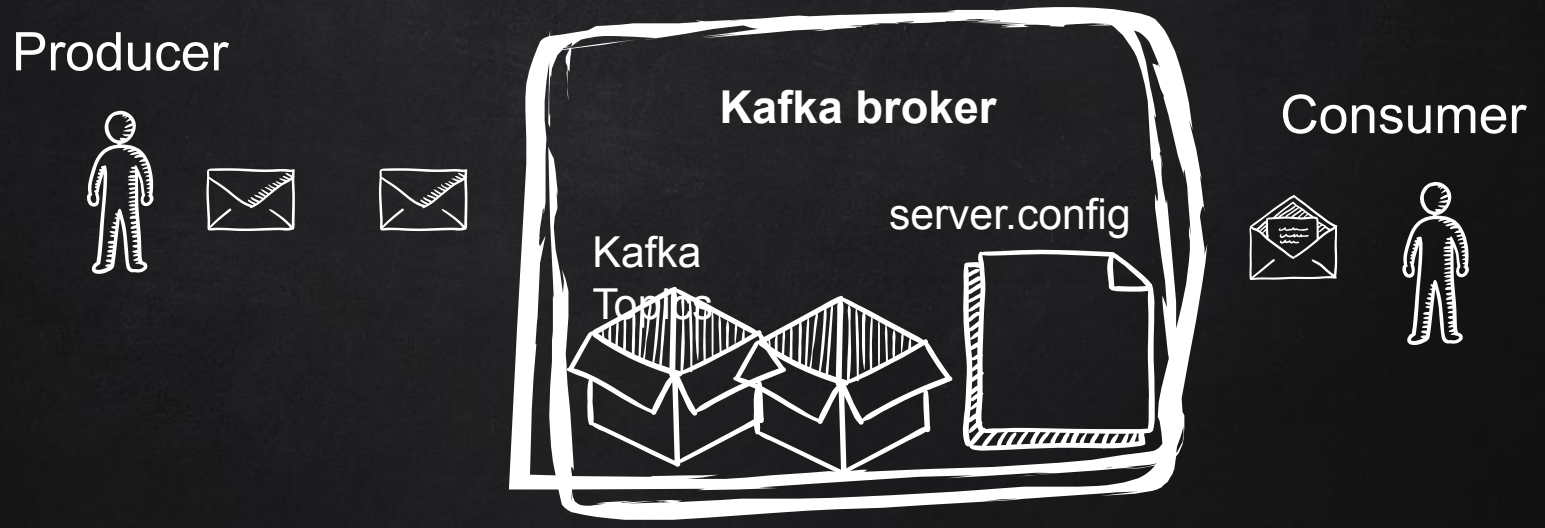
This does not help…
So let’s move to basics of the Kafka…

distributed even

it log service

Apache Kafka

Apache Kafka



Apache Kafka

Follower

Consumer groups

Kafka Mirror Maker

Leader election

Quorum

Preferred leader

Replication factor

KRaft

Controller nodes

ZooKeeper-based Kafka

Kafka Streams

Kafka Connect

And more…

Follov

Leader ele

KRaft

Kafka Strear

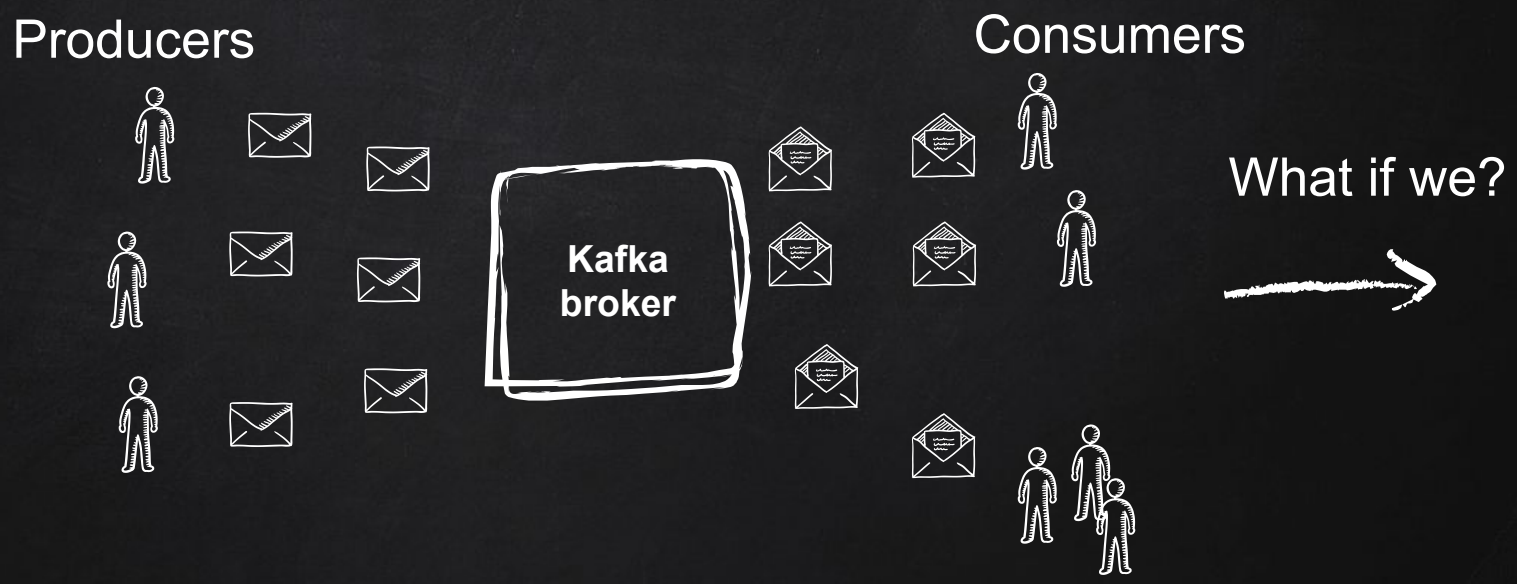
afka Mirror Maker

ader

ed Kafka

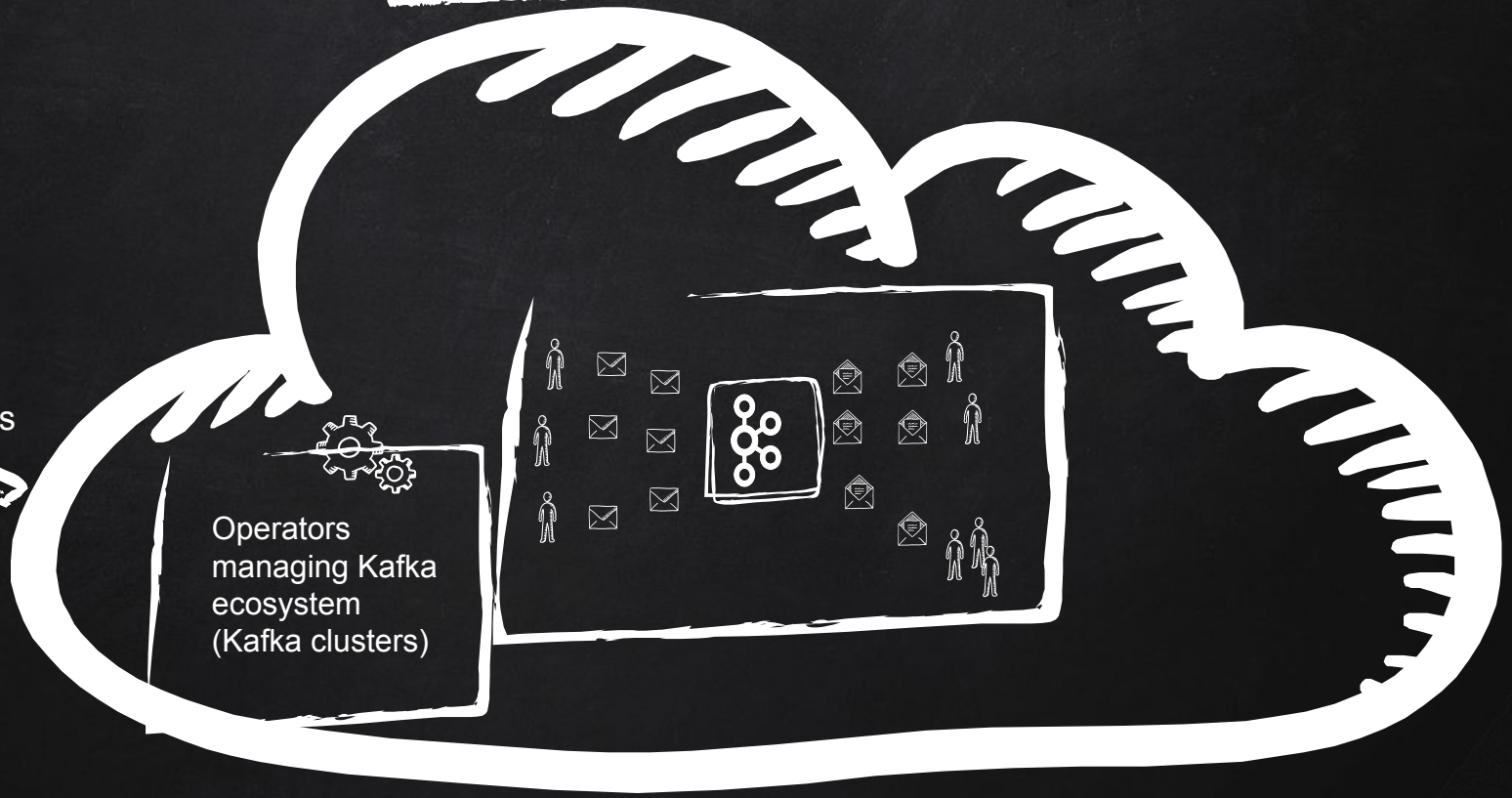
more…

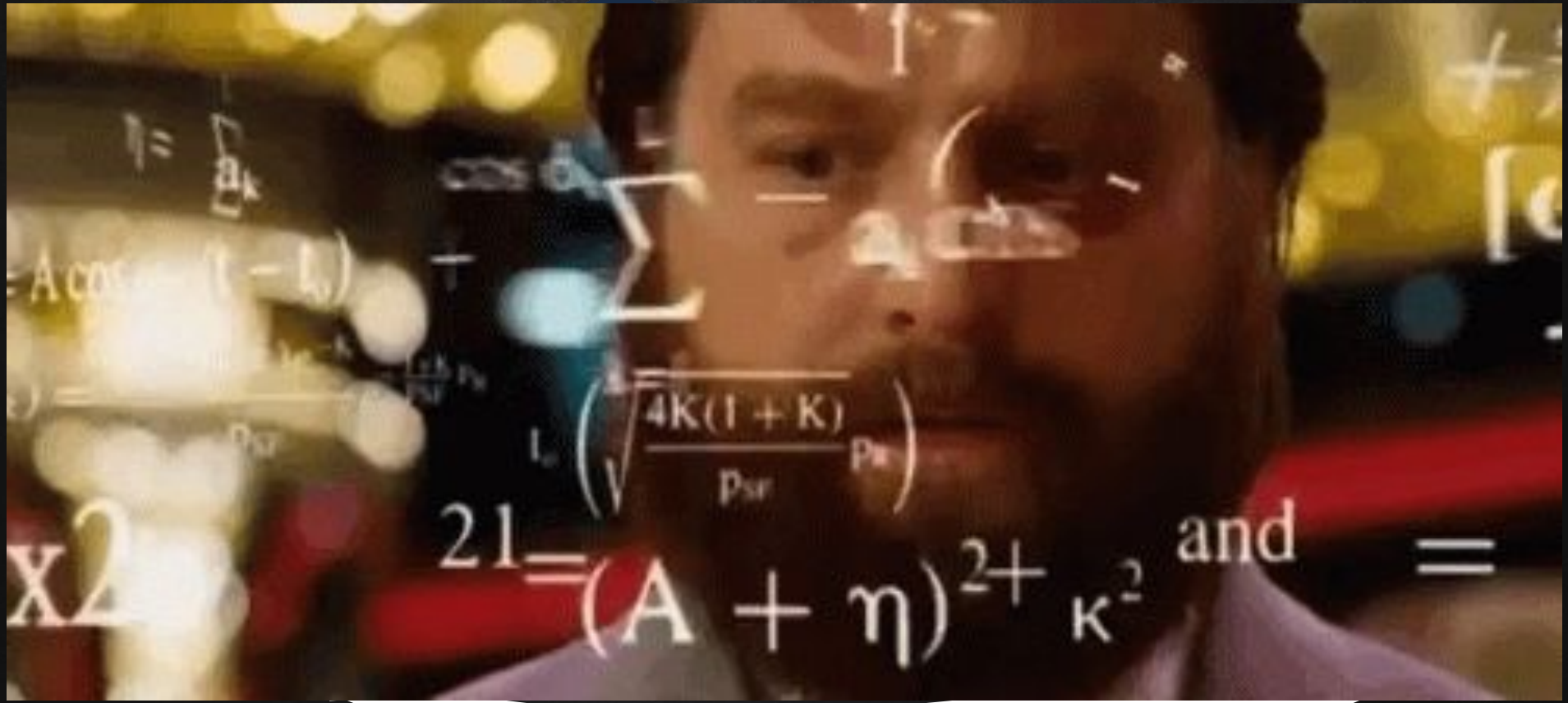
Apache Kafka



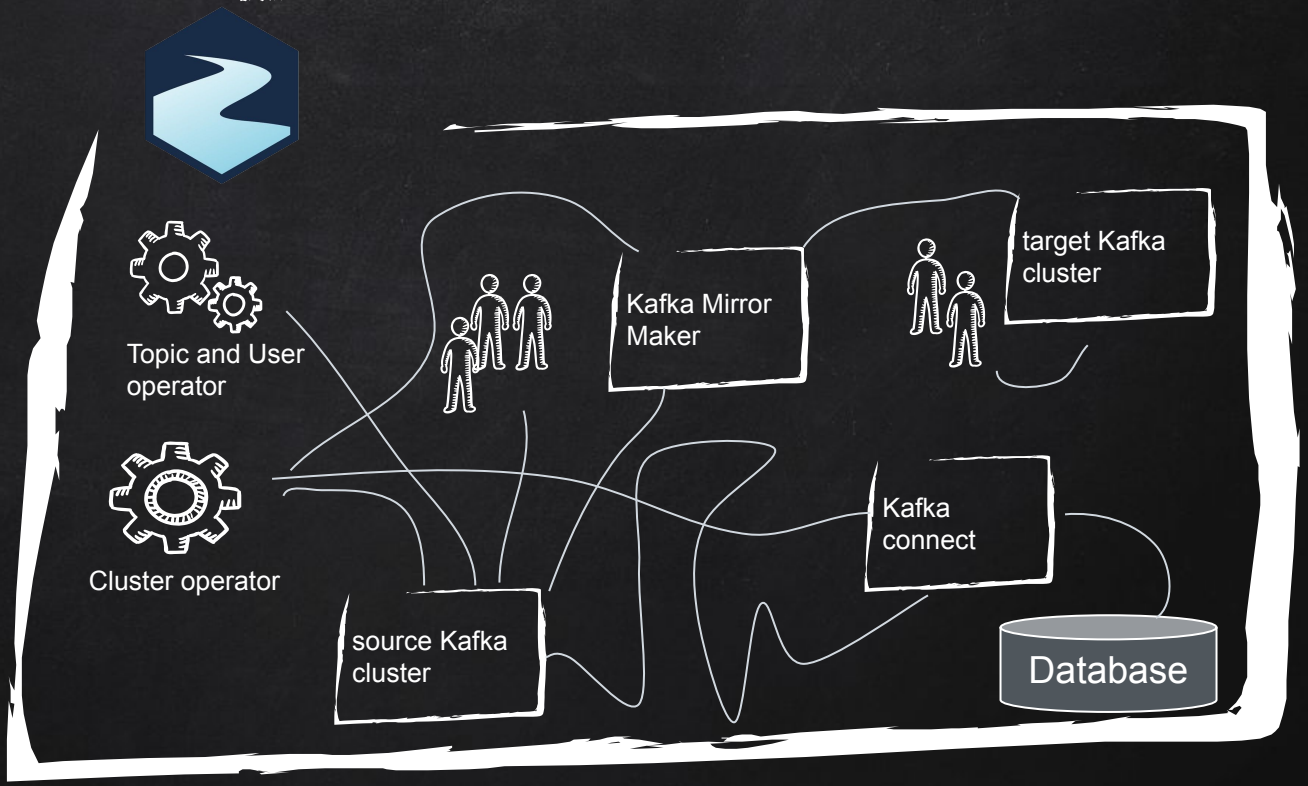
Strimzi

encapsulate
such system
in Kubernetes




21 = (A + η)² + κ² and

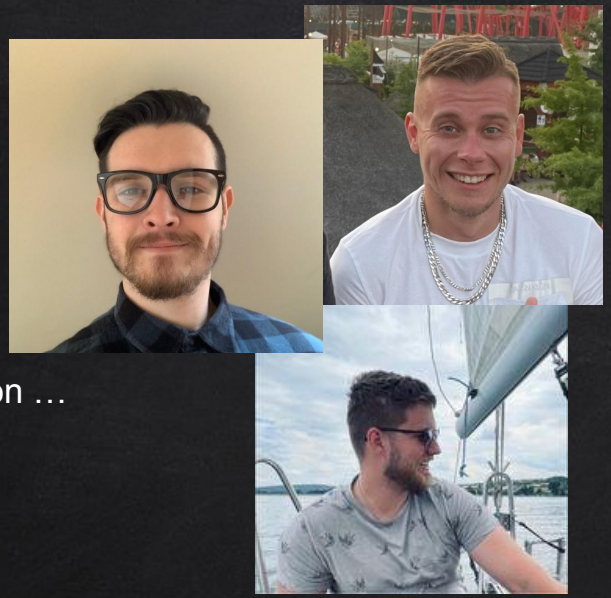
Strimzi



Complexity is really high


NETFLIX

skodjob



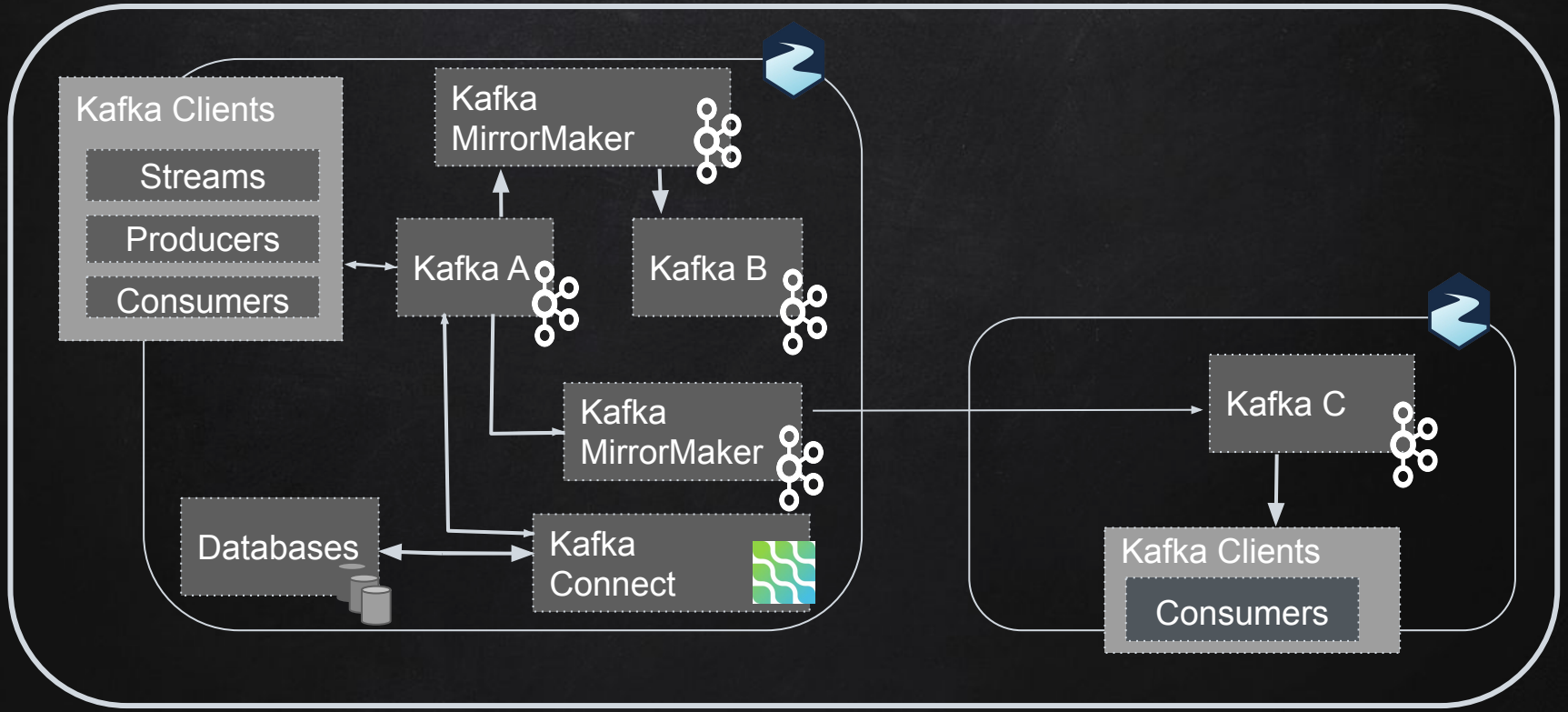
… to the production …

thanks to these guys we are able to run Strimzi in testing production environment…

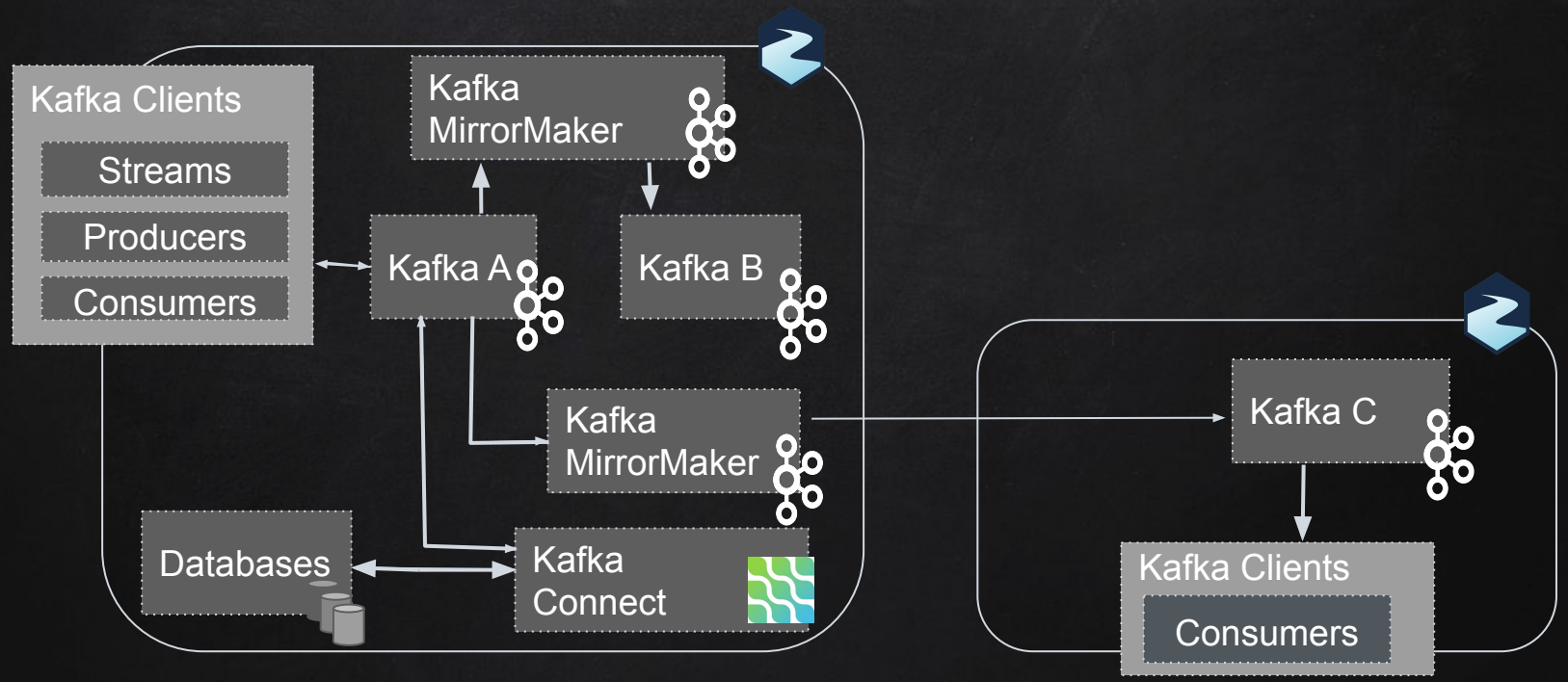
Production environment for Strimzi and other projects.

skodjob

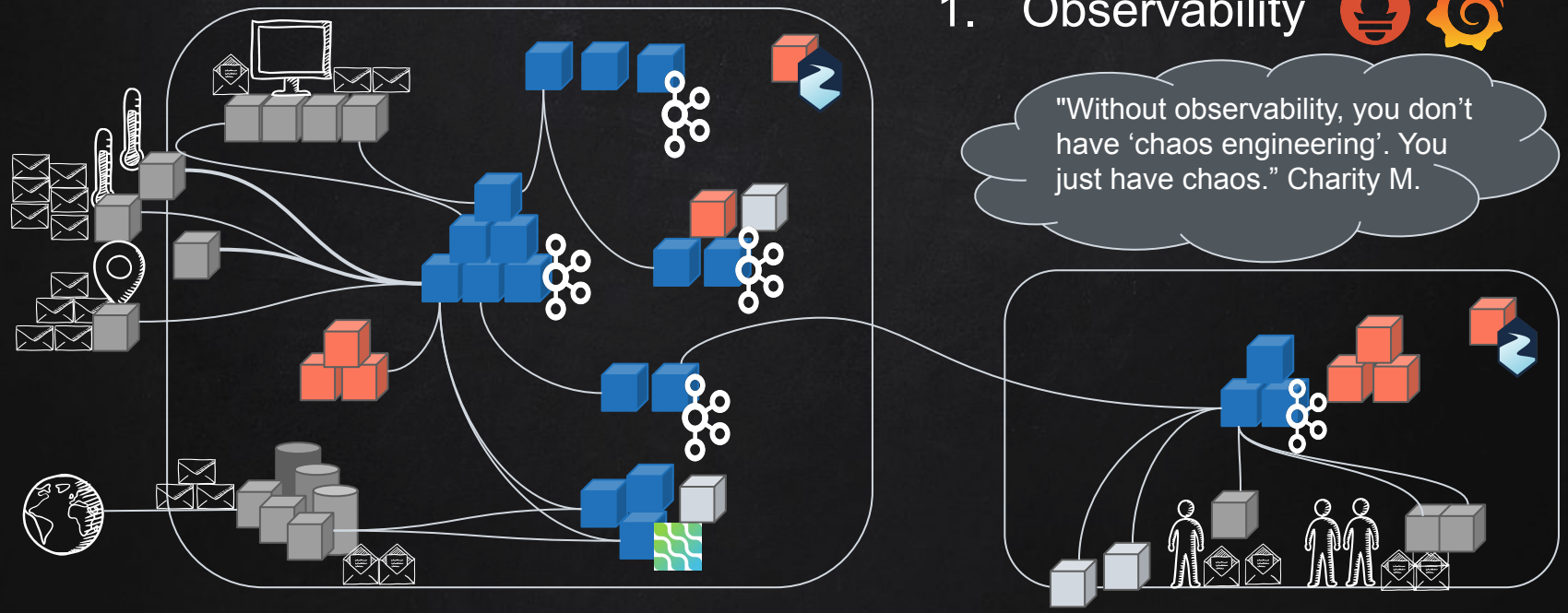

ANSIBLE
TEKTON
argo


Kafka Clients
Streams
Producers
Consumers
Kafka MirrorMaker
Kafka A
Kafka B
Kafka MirrorMaker
Databases
Kafka Connect
Kafka C
Kafka Clients
Consumers

Chaos Experiment - Intuition



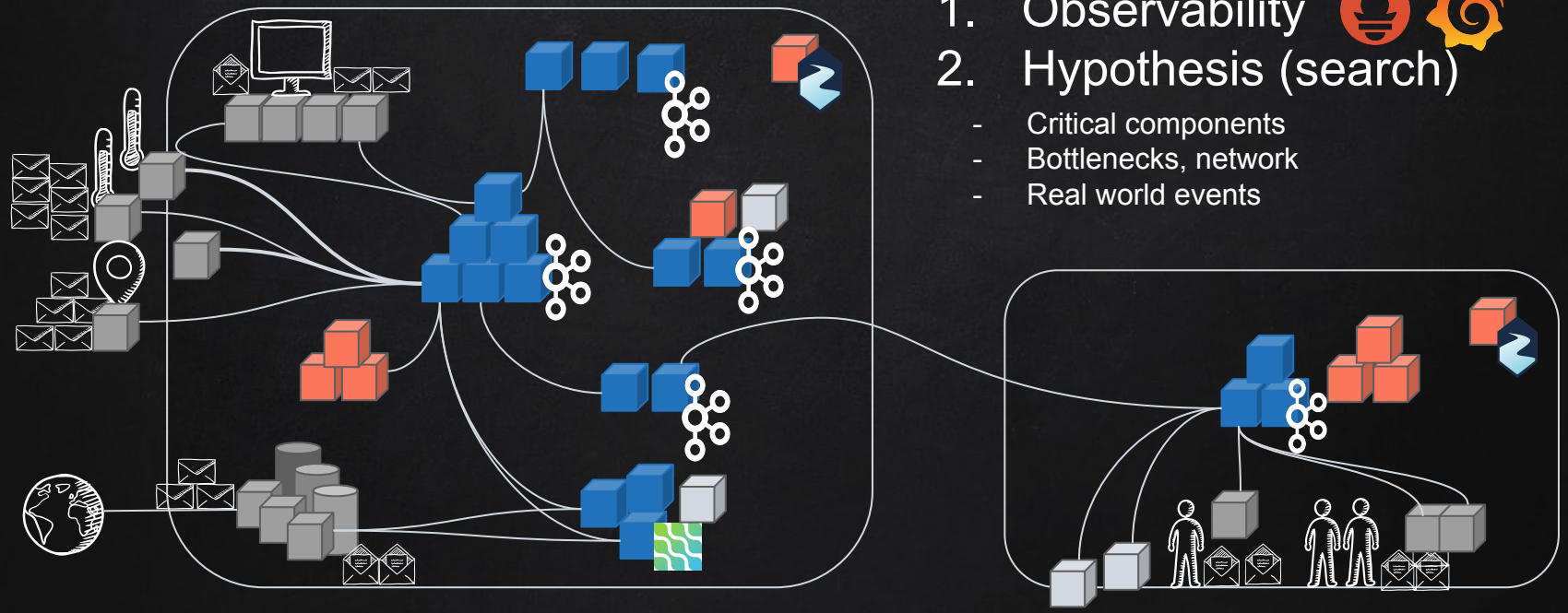
Chaos Experiment - Intuition



1. Observability

"Without observability, you don't have 'chaos engineering'. You just have chaos." Charity M.

Chaos Experiment - Hypothesis



1. Observability
2. Hypothesis (search)
   - Critical components
   - Bottlenecks, network
   - Real world events

Chaos Experiment - Document

Component

Config:

- - - - -
- - -

Variables/Param:

- - - - - - -
- - - - - - -

Chaos:

- - - - -
- - - - - - - - -

Notes:

- - - - - -
- - - - - - - -

Kafka

Config:

- Kraft
- Zookeeper

Params:

- Replicas
- Global configs
- Ephemeral vs Persistent

Chaos:

- Pod
- Network

Topic Operator

Config:

- Unidirectional
- Bidirectional

Chaos:

- Network …

Containers

Notes:

- m containers per pod in …
- JVM

Clients

Config:

- Producer
- Consumer
- Streams, Http

Chaos:

- Http, Network, DNS

Params:

- acks
- retries
- connections.max .idle

Bridge

Chaos:

- Http
- Network
- Pod

Notes:

- Http client only

Infrastructure

Chaos:

- DNS
- Network
- Node

Mirror Maker & Kafka Connect

Params:

- Different DBs (MySQL, Mongo …)
- Connector Type, Tasks/workers

Zookeeper & Kraft

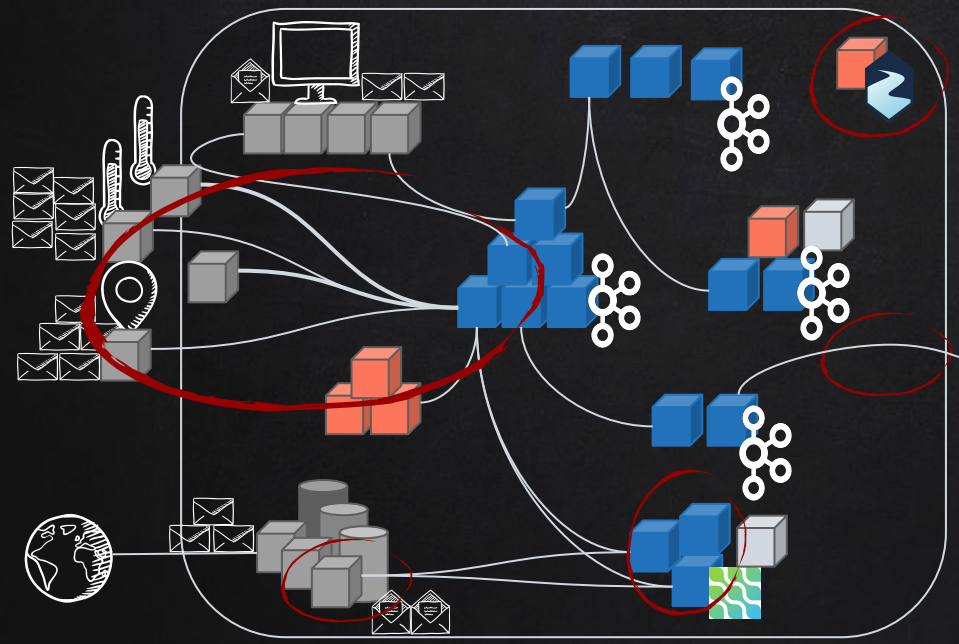
Params:

- Quorum necessary
- metadata

Others

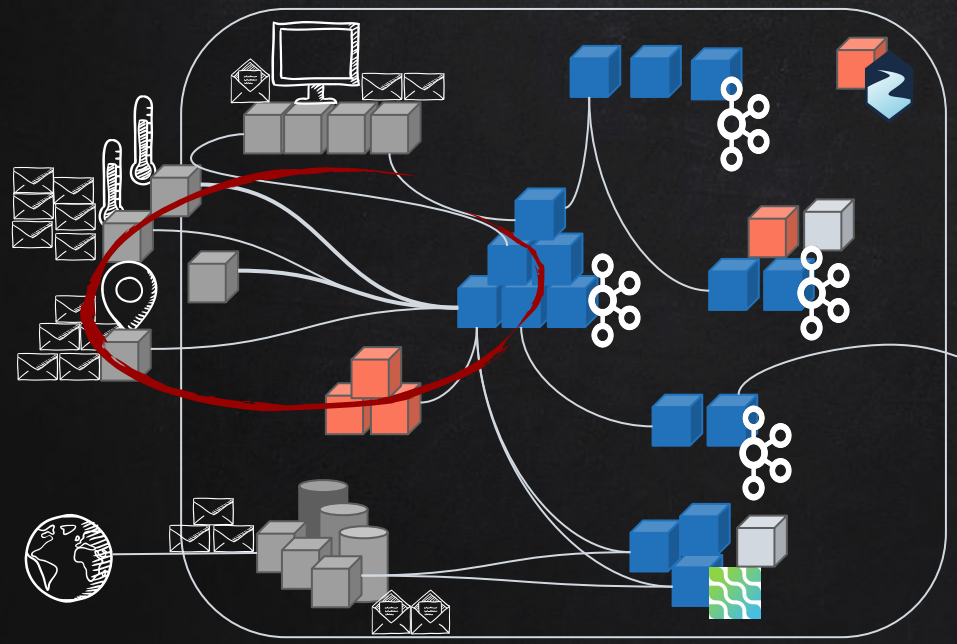
- …

Chaos Experiment - Hypothesis



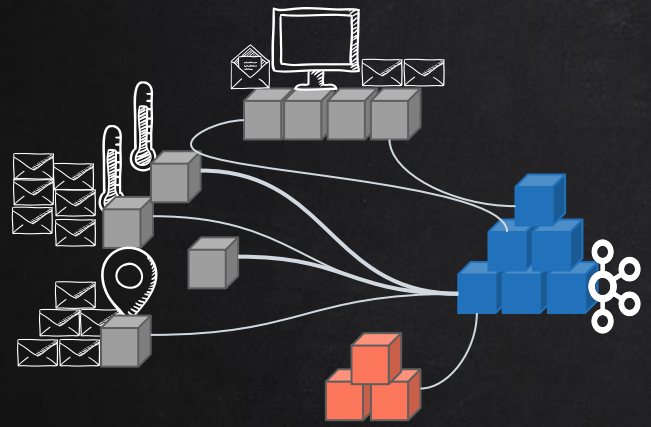
1. Observability
2. Hypothesis (search)
   - Critical components
   - Bottlenecks, network
   - Real world events

Chaos Experiment - Hypothesis



1. Observability
2. Hypothesis (formulate)

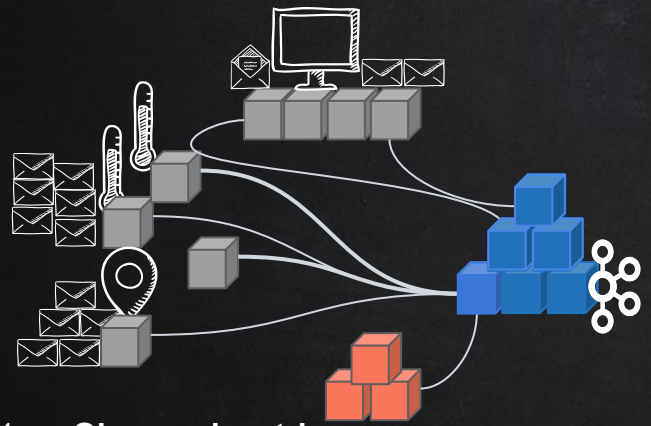
Chaos Experiment - Hypothesis



1. Observability
2. Hypothesis (formulate)

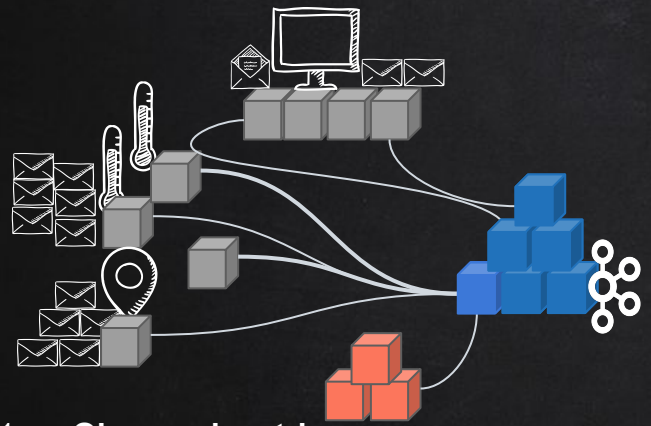
Chaos Experiment - Hypothesis

1. Observability
2. Hypothesis (formulate)

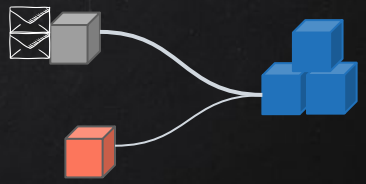


1. **Observed metrics:**
   a. Incoming traffic metrics
   b. Ready brokers
   c. CPU used, memory
2. **Hypothese:** (Production system) Kafka cluster can withstand failure of 3 brokers without loss of messages or cascading fails.

Chaos Experiment - Scale



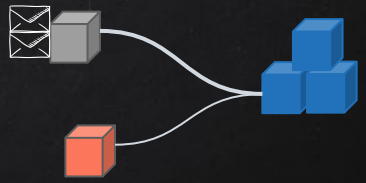
1. Observability
2. Hypothesis
3. Scale (down & up)



1. **Observed metrics:**
   a. Incoming traffic metrics
   b. Ready brokers
   c. CPU used, memory
2. **Hypothese:** (Production system) Kafka cluster can withstand failure of 3 brokers without loss of messages or cascading fails.

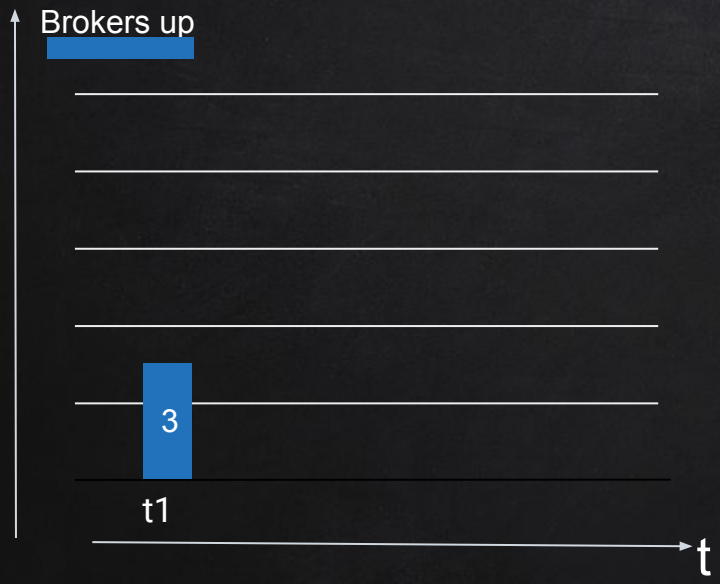
Chaos Experiment - Timeline

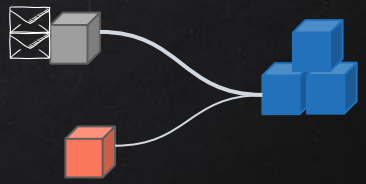
1. Observability
2. Hypothesis
3. Scale



Chaos Experiment - Timeline

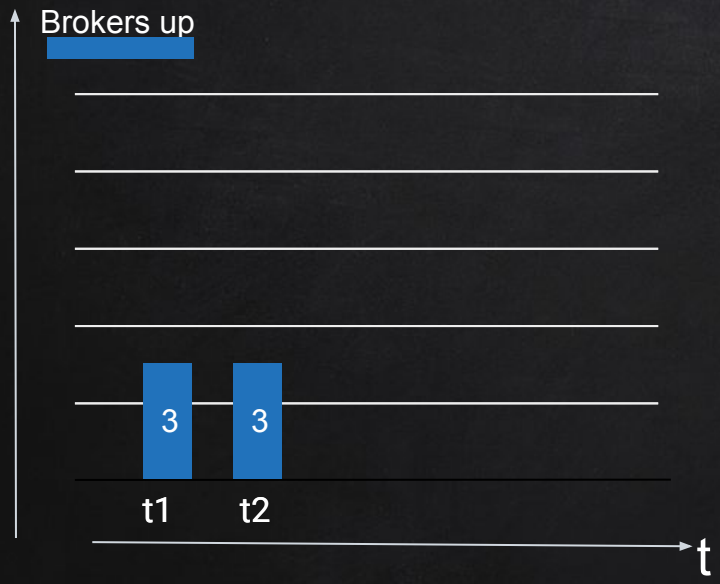
1. Observability
2. Hypothesis
3. Scale

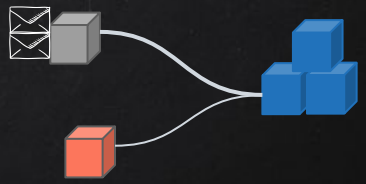




Chaos Experiment - Timeline

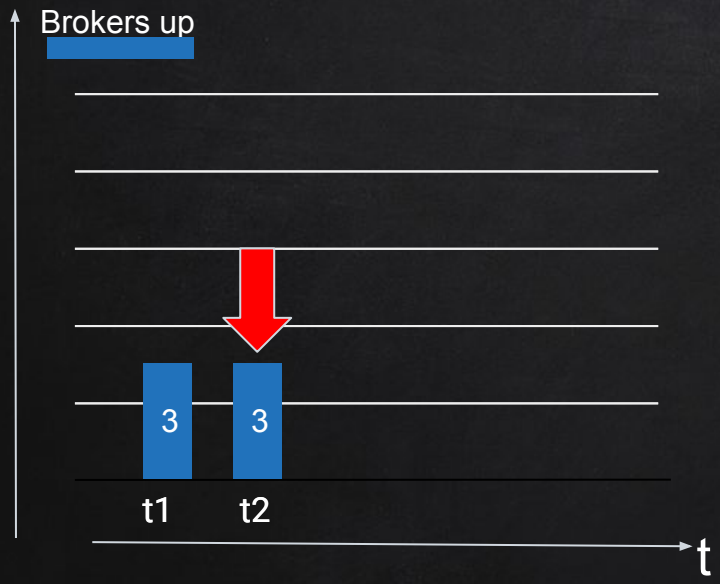
1. Observability
2. Hypothesis
3. Scale

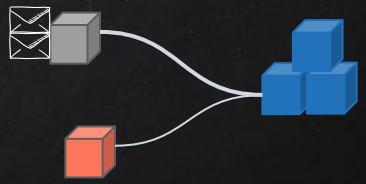




Chaos Experiment - Timeline

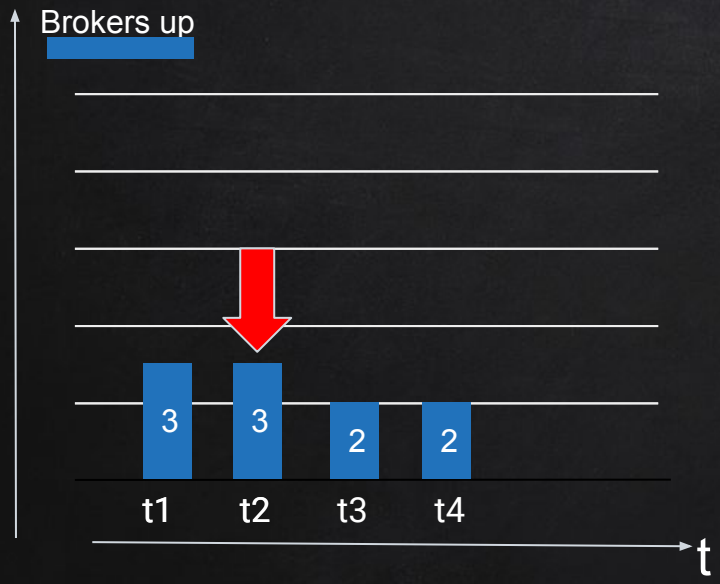
1. Observability
2. Hypothesis
3. Scale

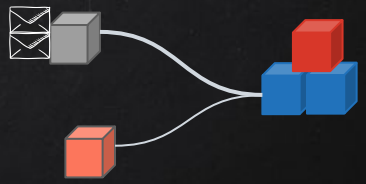




Chaos Experiment - Timeline

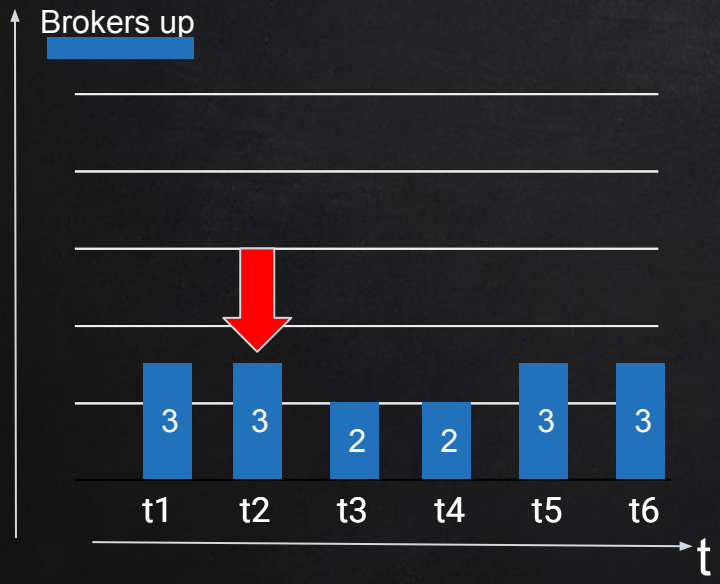
1. Observability
2. Hypothesis
3. Scale

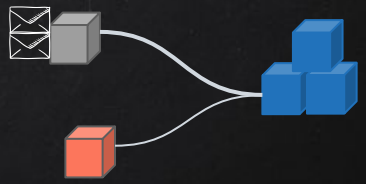




Chaos Experiment - Timeline

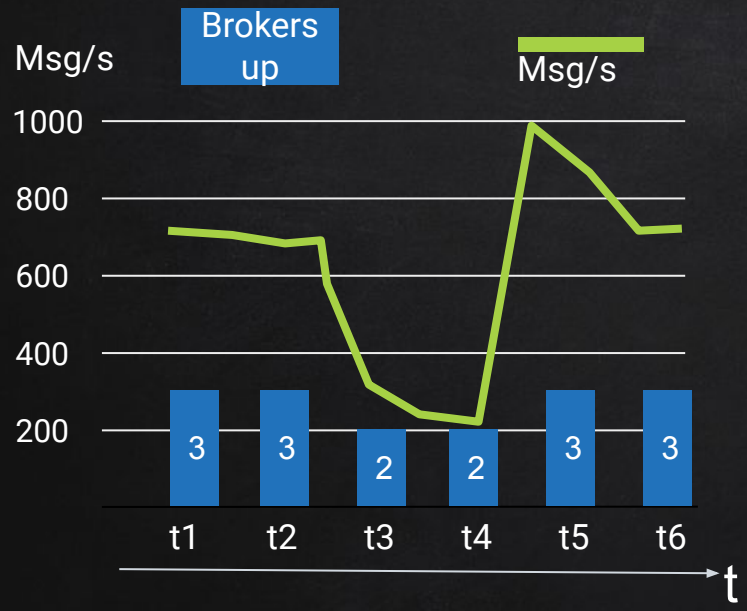
1. Observability
2. Hypothesis
3. Scale

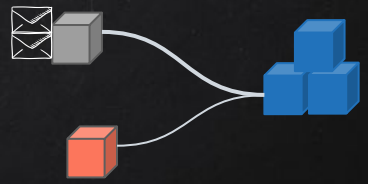




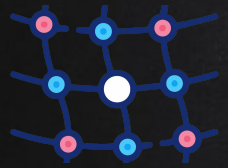
Chaos Experiment - Scale

1. Observability
2. Hypothesis
3. Scale

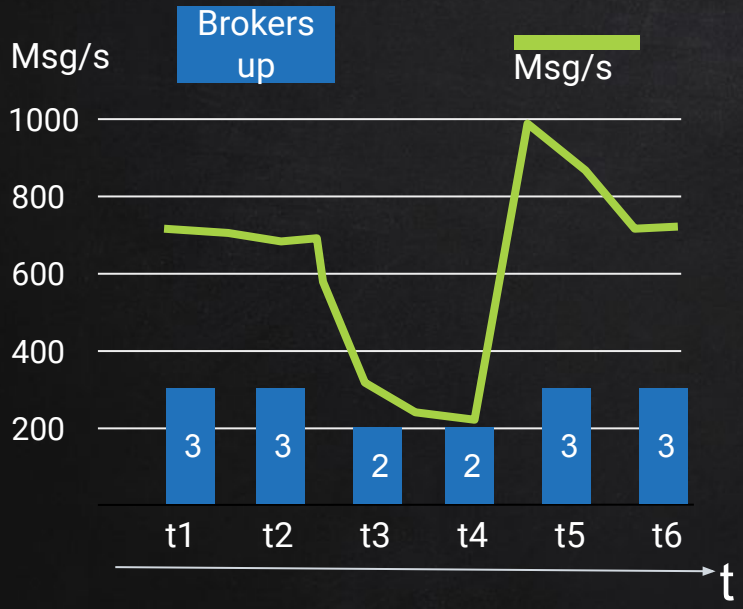


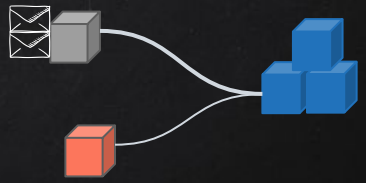


Chaos Experiment - Tools

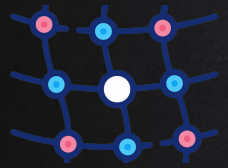


1. Observability
2. Hypothesis
3. Scale
4. Run & Results

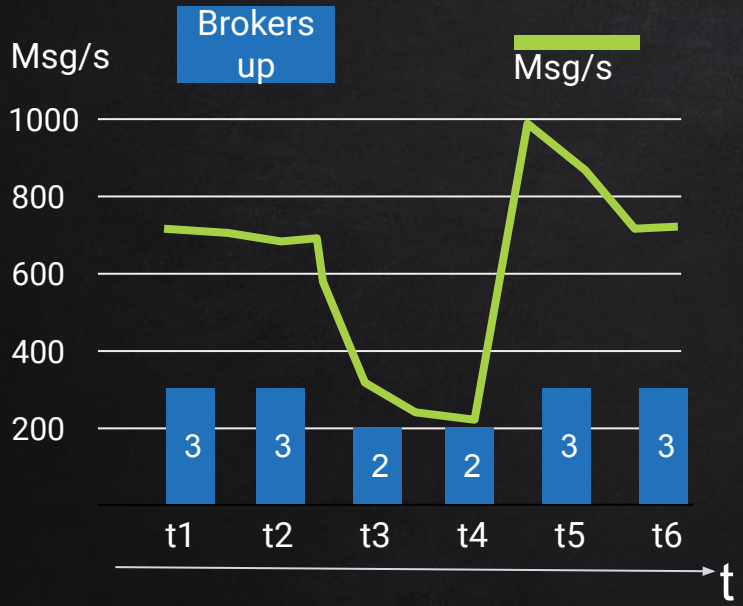


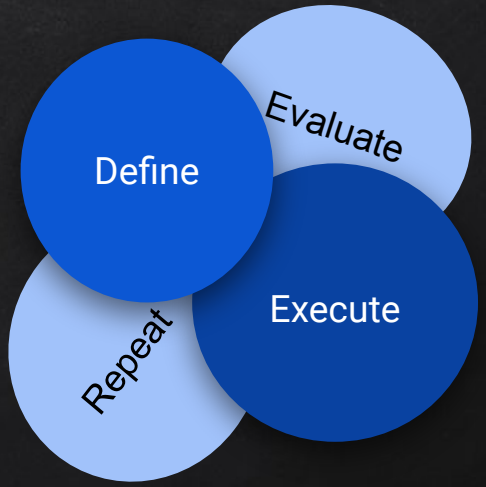


Chaos Experiment - Tools

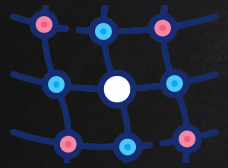


1. Observability
2. Hypothesis
3. Scale
4. Run & Results

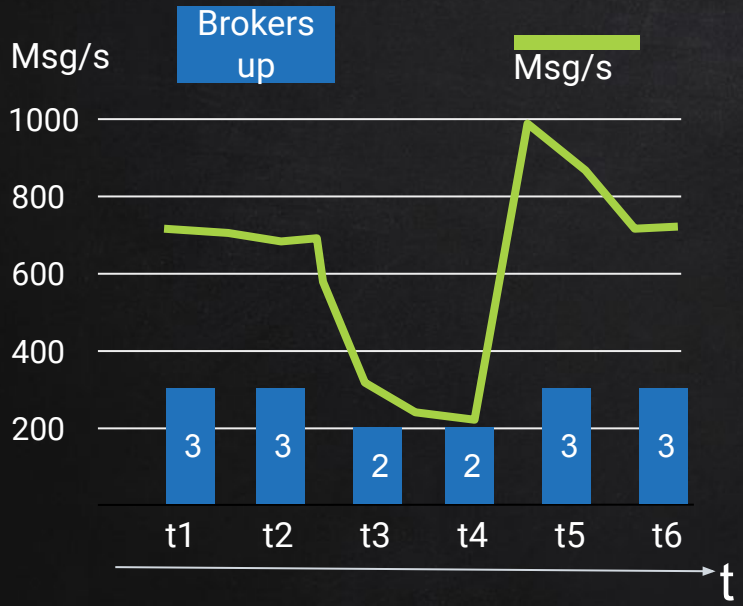


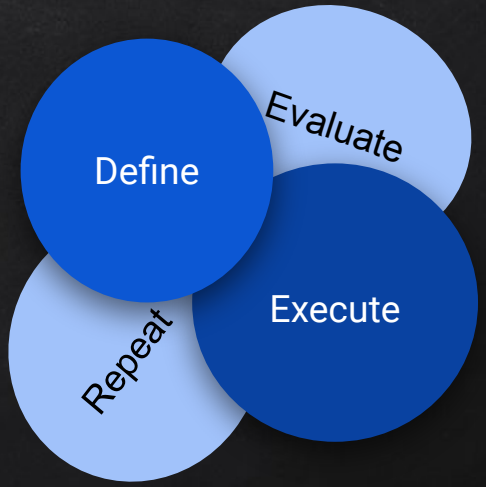


Chaos Experiment - Tools



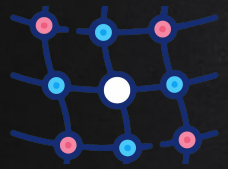
1. Observability
2. Hypothesis
3. Scale
4. Run & Results



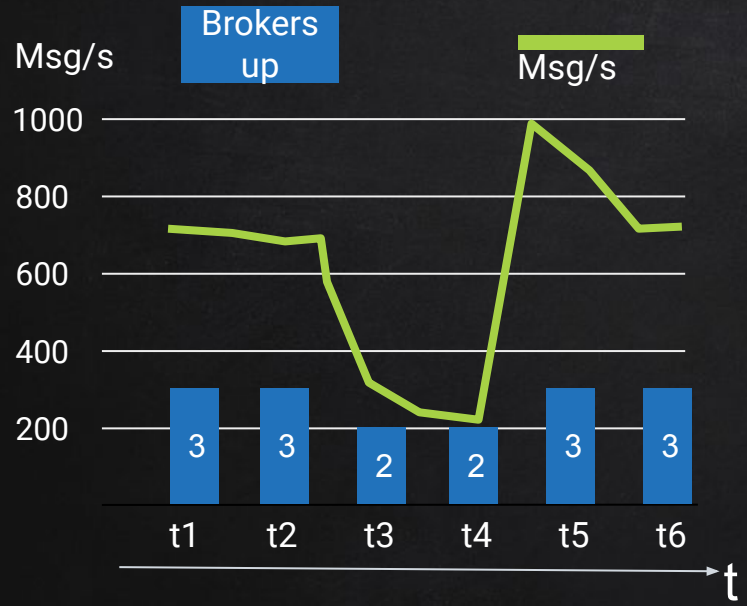


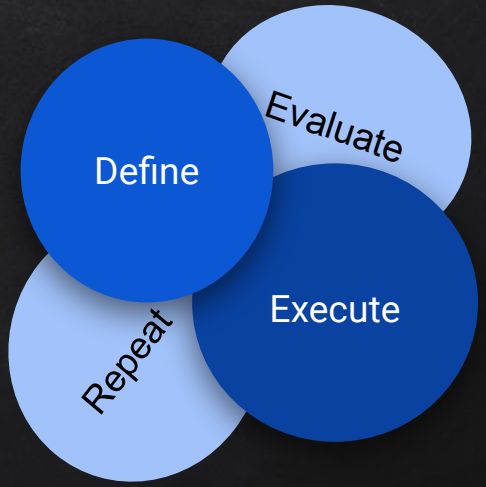

```
kind: PodChaos
metadata:
 name: broker-kill-66
spec:
 action: pod-kill
 mode: one
 selector:
   namespaces:
     - kafka-main
   labelSelectors:
. . .
```

Chaos Experiment - Tools



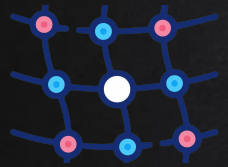
1. Observability
2. Hypothesis
3. Scale
4. Run & Results



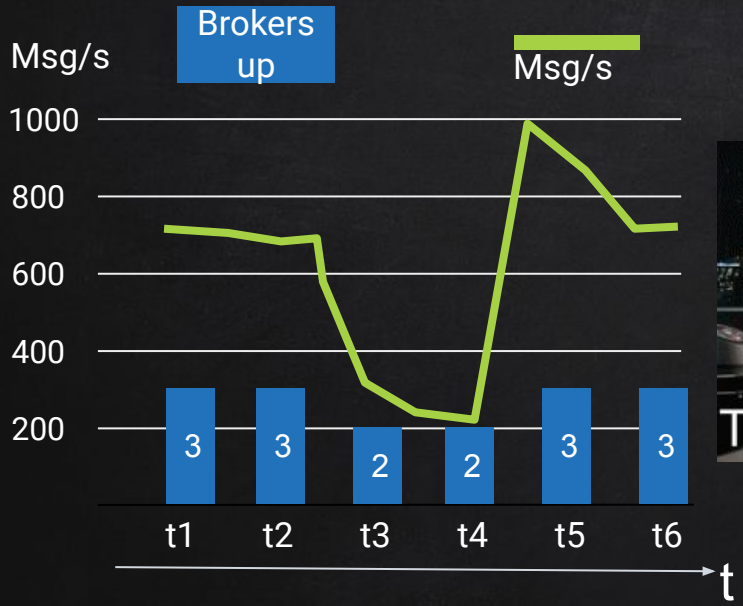



```
kind: PodChaos
metadata:
 name: broker-kill-66
spec:
 action: pod-kill
 mode: one
 selector:
   namespaces:
     - kafka-main
   labelSelectors:
. . .
```

Chaos Experiment - Run



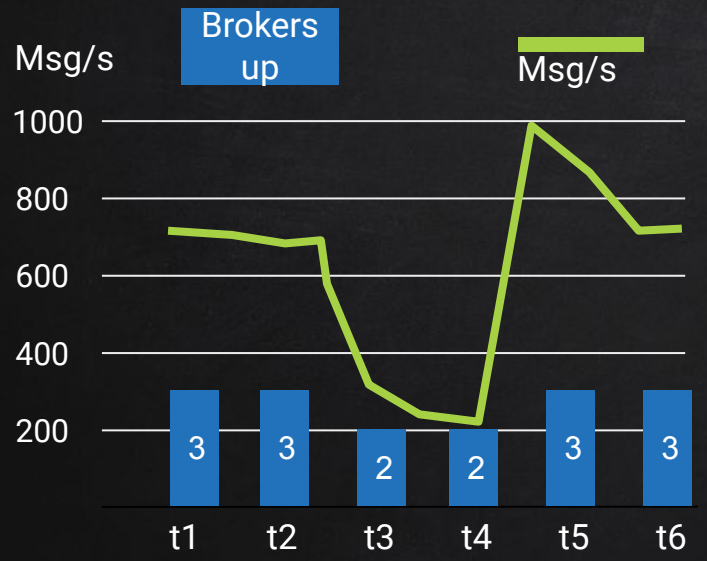
1. Observability
2. Hypothesis
3. Scale
4. Run & Results




```
kind: PodChaos
metadata:
 name: broker-kill-66
spec:
 action: pod-kill
 mode: one
 selector:
   namespaces:
     - kafka-main
   labelSelectors:
. . .
```

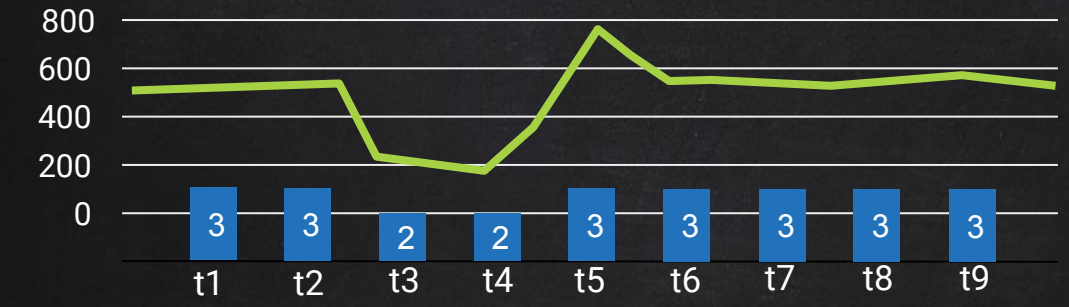
Chaos Experiment - Results

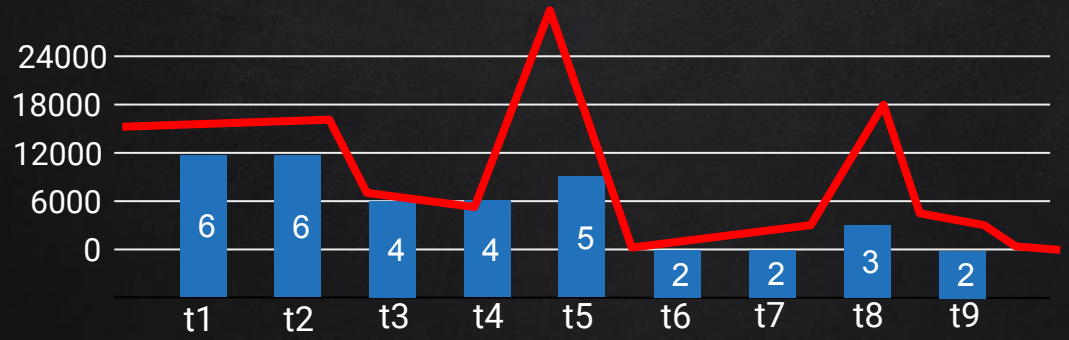
type	Traffic_in (msg/s)	replicas_down_to	Duration (m)	result
chaos-66	650 (670 base)	2/3	6	✅




```
kind: PodChaos
metadata:
 name: broker-kill-66
spec:
 action: pod-kill
 mode: one
 selector:
   namespaces:
     - kafka-main
   labelSelectors:
. . .
```

Chaos Experiment - Repeat





Demo I: Broker(s) failure

Description: Fail of critical component Pod(s).

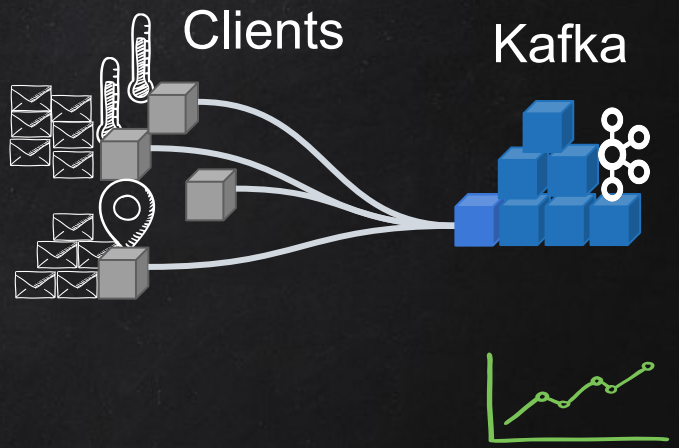
Observability: Ensure the availability of metrics for CPU, memory, and traffic in Kafka Pods.

Demo I: Broker(s) failure

Description: Fail of critical component Pod(s).

Observability: Ensure the availability of metrics for CPU, memory, and traffic in Kafka Pods.

Steady State: All broker and client replicas are up and ready, with communication throughput stable and free of spikes



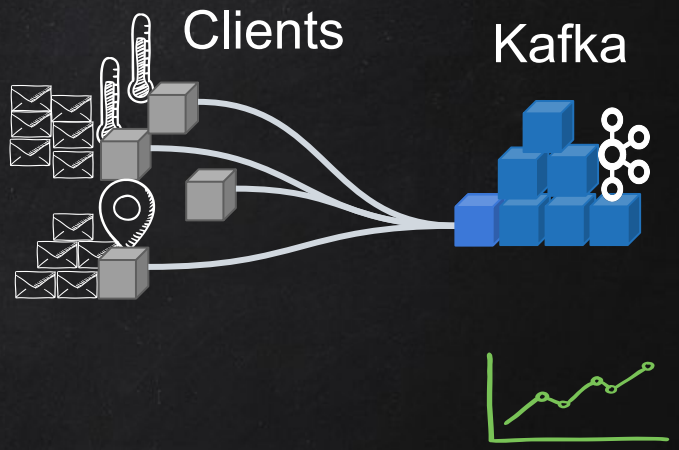
Demo I: Broker(s) failure

Description: Fail of critical component Pod(s).

Observability: Ensure the availability of metrics for CPU, memory, and traffic in Kafka Pods.

Steady State: All broker and client replicas are up and ready, with communication throughput stable and free of spikes

Hypothesis: **Eliminating three out of seven brokers will not result in cascading failures, and user impact will be minimal. Throughput may significantly decrease but should not drop to zero, and the disruption should not last longer than the time required to respawn lost instances (approximately 1 minute).**



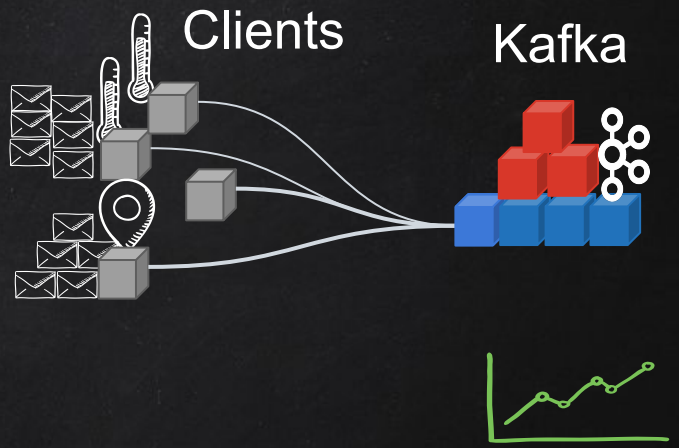
Demo I: Broker(s) failure

Description: Fail of critical component Pod(s).

Observability: Ensure the availability of metrics for CPU, memory, and traffic in Kafka Pods.

Steady State: All broker and client replicas are up and ready, with communication throughput stable and free of spikes

Hypothesis: **Eliminating three out of seven brokers will not result in cascading failures, and user impact will be minimal. Throughput may significantly decrease but should not drop to zero, and the disruption should not last longer than the time required to respawn lost instances (approximately 1 minute).**



Demo I: Broker(s) failure

Description: Fail of critical component Pod(s).

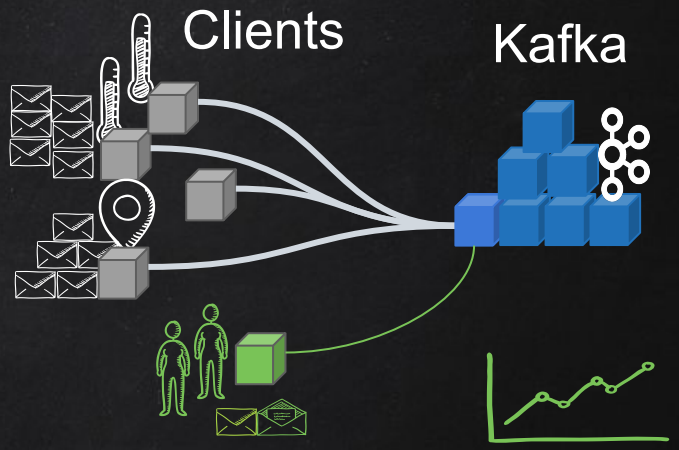
Observability: Ensure the availability of metrics for CPU, memory, and traffic in Kafka Pods.

Steady State: All broker and client replicas are up and ready, with communication throughput stable and free of spikes

Hypothesis: **Eliminating three out of seven brokers will not result in cascading failures, and user impact will be minimal. Throughput may significantly decrease but should not drop to zero, and the disruption should not last longer than the time required to respawn lost instances (approximately 1 minute).**

Checks:

- All Kafka Pods Ready
- All produced messages consumed



```
Handling connection for 9090
Handling connection for 9090
```

```
~/Documents/Work/StrimKKhaos git:(main) ✗ 5
```

```
Every 2.0s: oc get pod --selector strimzi.io/broker-role="true...  morsak-mac: Fri Feb  2 16:15:38 2024

NAME                  READY   STATUS    RESTARTS         AGE
my-cluster-kafka-0    1/1     Running   10 (4m58s ago)   28h
my-cluster-kafka-1    1/1     Running   7 (21m ago)      28h
my-cluster-kafka-2    1/1     Running   11 (5m17s ago)   28h
my-cluster-kafka-3    1/1     Running   2 (34m ago)      3h58m
my-cluster-kafka-4    1/1     Running   6 (157m ago)     3h58m
my-cluster-kafka-5    1/1     Running   12 (5m18s ago)   3h58m
my-cluster-kafka-6    1/1     Running   4 (34m ago)      3h58m
```

```
Every 2.0s: oc get nodes                                  morsak-mac: Fri Feb  2 16:15:39 2024

NAME                           STATUS   ROLES                  AGE   VERSION
majk-414-wvkf9-master-0        Ready    control-plane,master   29d   v1.27.8+4fab27b
majk-414-wvkf9-master-1        Ready    control-plane,master   29d   v1.27.8+4fab27b
majk-414-wvkf9-master-2        Ready    control-plane,master   29d   v1.27.8+4fab27b
majk-414-wvkf9-worker-0-g65cf  Ready    worker                 29d   v1.27.8+4fab27b
majk-414-wvkf9-worker-0-jfrkd  Ready    worker                 29d   v1.27.8+4fab27b
majk-414-wvkf9-worker-0-t7mt8  Ready    worker                 28d   v1.27.8+4fab27b
```

```
Every 2.0s: oc get pod --selector app=java-kafka-producer -n m...  morsak-mac: Fri Feb  2 16:15:37 2024

NAME                                   READY   STATUS    RESTARTS   AGE
java-kafka-producer-5dccd56768-97kp5   1/1     Running   0          16m
java-kafka-producer-5dccd56768-b4qdp   1/1     Running   0          16m
java-kafka-producer-5dccd56768-bmcpl   1/1     Running   0          16m
java-kafka-producer-5dccd56768-cxz8p   1/1     Running   0          16m
java-kafka-producer-5dccd56768-d8tgl   1/1     Running   0          15m
java-kafka-producer-5dccd56768-d9ck2   1/1     Running   0          15m
java-kafka-producer-5dccd56768-kpb9v   1/1     Running   0          15m
```

Demo II: Worker node crash

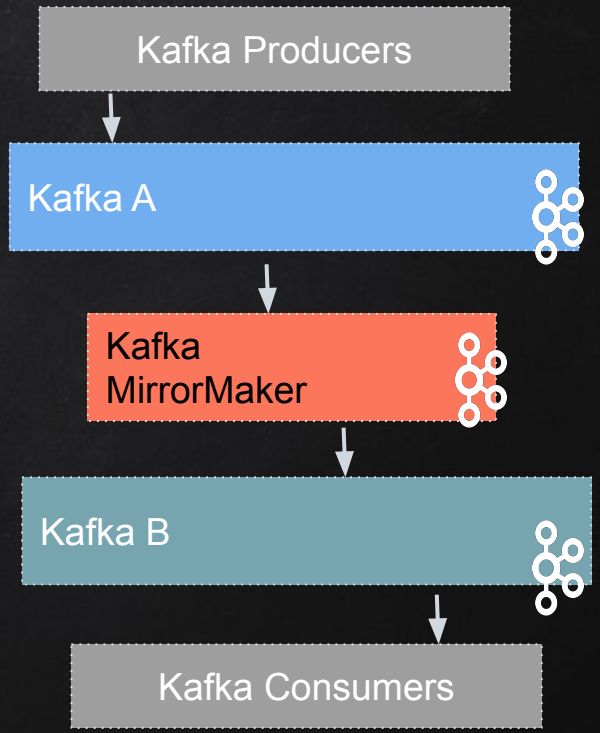
Description: Crash of a worker node effect on services (Kafkas and Mirror Maker).

Observability: Ensure the availability of all services across the cluster; monitor for any unexpected events within the cluster.

Demo II: Worker node crash

Description: Crash of a worker node effect on services (Kafkas and Mirror Maker).

Observability: Ensure the availability of all services across the cluster; monitor for any unexpected events within the cluster.

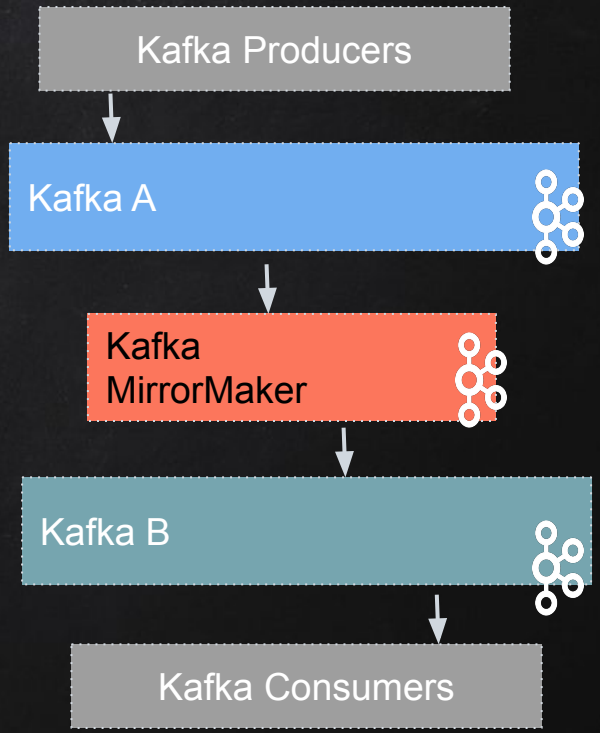


Demo II: Worker node crash

Description: Crash of a worker node effect on services (Kafkas and Mirror Maker).

Observability: Ensure the availability of all services across the cluster; monitor for any unexpected events within the cluster.

Steady State: All services are fully available and ready to accept traffic.



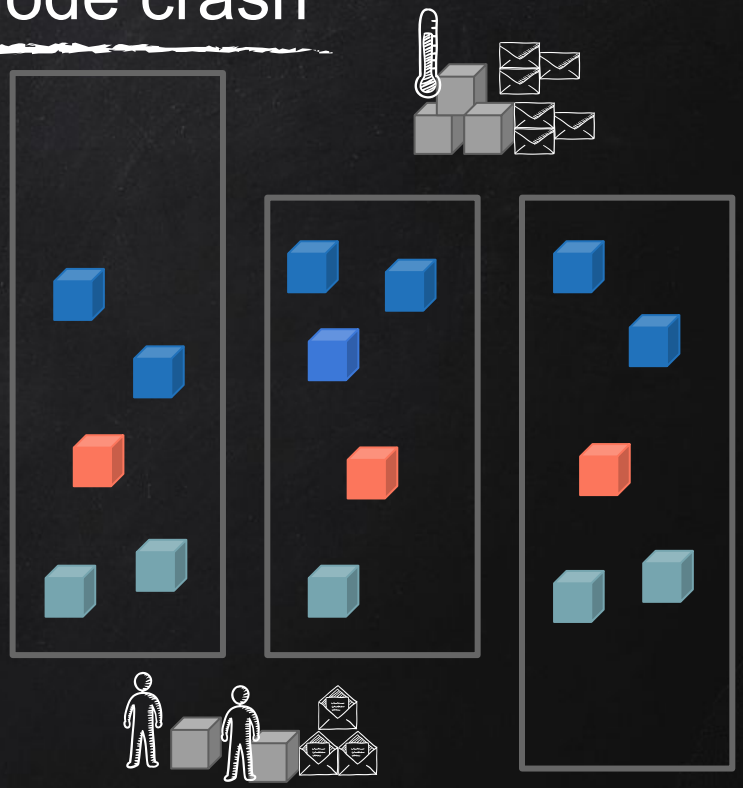
Demo II: Worker node crash

Description: Crash of a worker node effect on services (Kafkas and Mirror Maker).

Observability: Ensure the availability of all services across the cluster; monitor for any unexpected events within the cluster.

Steady State: All services are fully available and ready to accept traffic.

Hypothesis: **Eliminating one of the Kubernetes worker pools will not bring down any services, even temporarily. The cluster will recover, and within a reasonable time period, all services will return to their full replica count.**



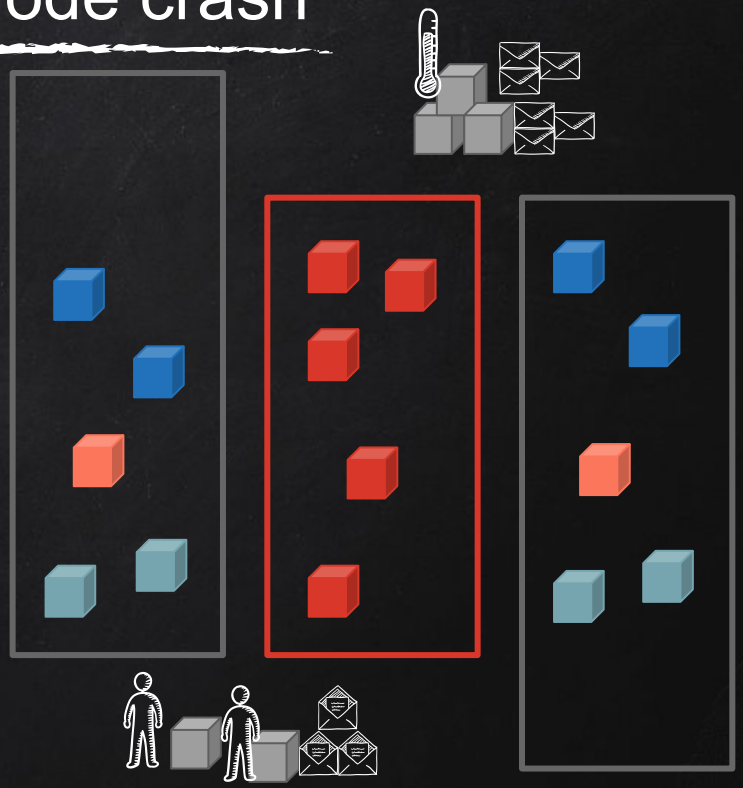
Demo II: Worker node crash

Description: Crash of a worker node effect on services (Kafkas and Mirror Maker).

Observability: Ensure the availability of all services across the cluster; monitor for any unexpected events within the cluster.

Steady State: All services are fully available and ready to accept traffic.

Hypothesis: **Eliminating one of the Kubernetes worker pools will not bring down any services, even temporarily. The cluster will recover, and within a reasonable time period, all services will return to their full replica count.**



Demo II: Worker node crash

Description: Crash of a worker node effect on services (Kafkas and Mirror Maker).

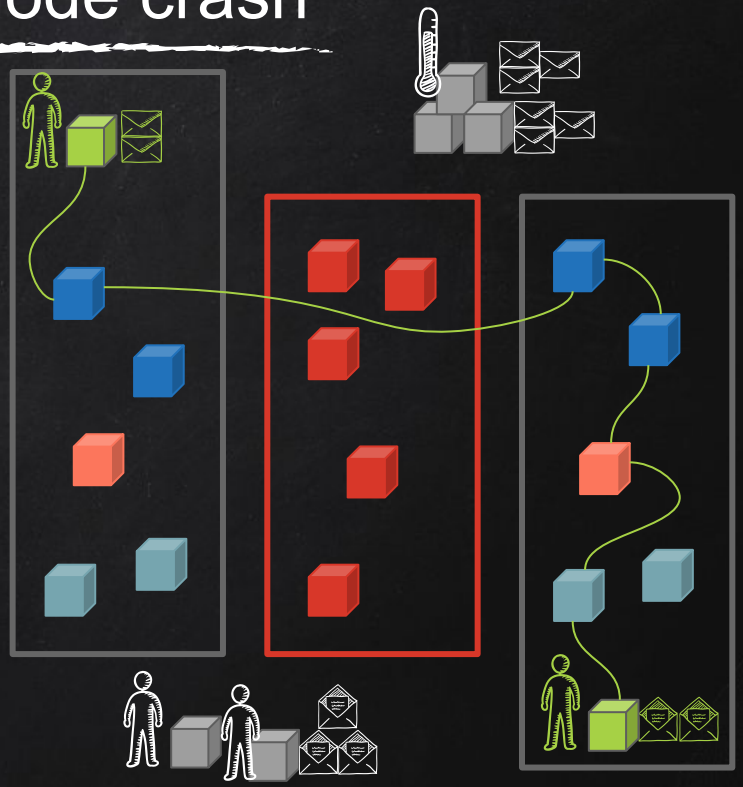
Observability: Ensure the availability of all services across the cluster; monitor for any unexpected events within the cluster.

Steady State: All services are fully available and ready to accept traffic.

Hypothesis: **Eliminating one of the Kubernetes worker pools will not bring down any services, even temporarily. The cluster will recover, and within a reasonable time period, all services will return to their full replica count.**

Checks:

- Verify that all Kafka clusters and accompanying services are ready.
- Ensure all messages produced are successfully consumed from the relevant clusters.



Demo II: Worker node crash

Description: Crash of a worker node effect on services (Kafkas and Mirror Maker).

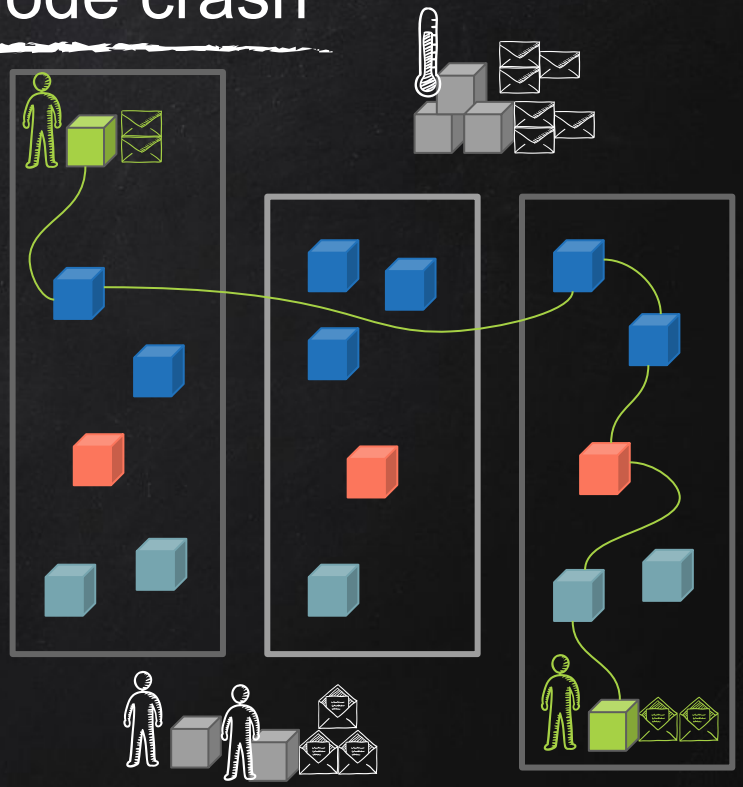
Observability: Ensure the availability of all services across the cluster; monitor for any unexpected events within the cluster.

Steady State: All services are fully available and ready to accept traffic.

Hypothesis: **Eliminating one of the Kubernetes worker pools will not bring down any services, even temporarily. The cluster will recover, and within a reasonable time period, all services will return to their full replica count.**

Checks:

- Verify that all Kafka clusters and accompanying services are ready.
- Ensure all messages produced are successfully consumed from the relevant clusters.



Embracing Chaos - Benefits

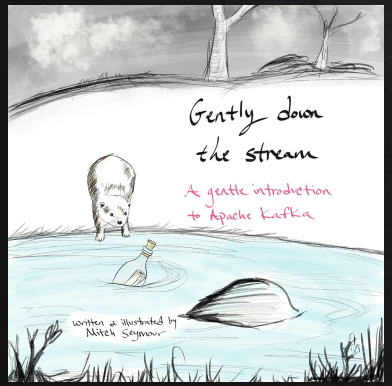
Embracing Chaos - Benefits

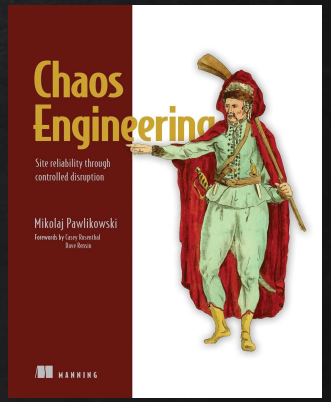
Embracing Chaos - How to ?

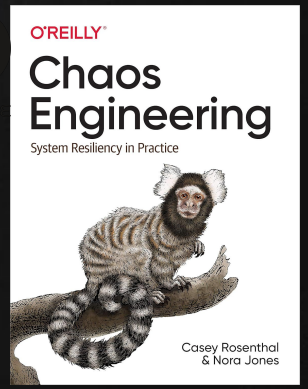
Embracing Chaos - How to ?

Embracing Chaos - How to ?

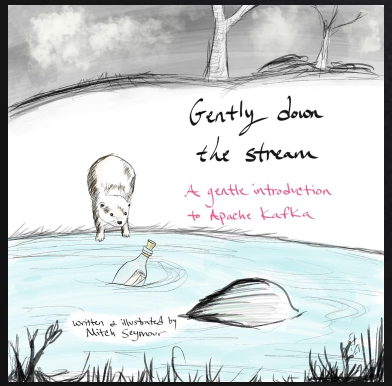
Embracing Chaos - How to ?

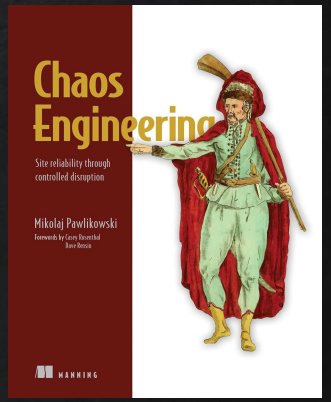


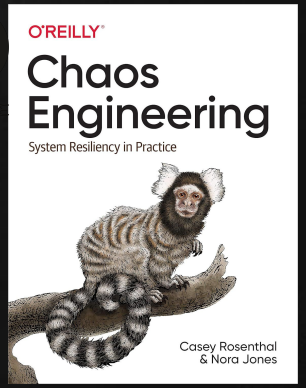




Embracing Chaos - How to ?







Thank you!