MODERNISATION EXAMPLES

Resilience testing

HUSTEF
HUNGARIAN SOFTWARE TESTING FORUM

ENDAVA
Živan Sinžar

Agenda

1. Introduction
2. Implementation ideas
3. Challenges
4. Conclusion
5. Questions

How we got here?

What is Resilience testing?

Why it is important?

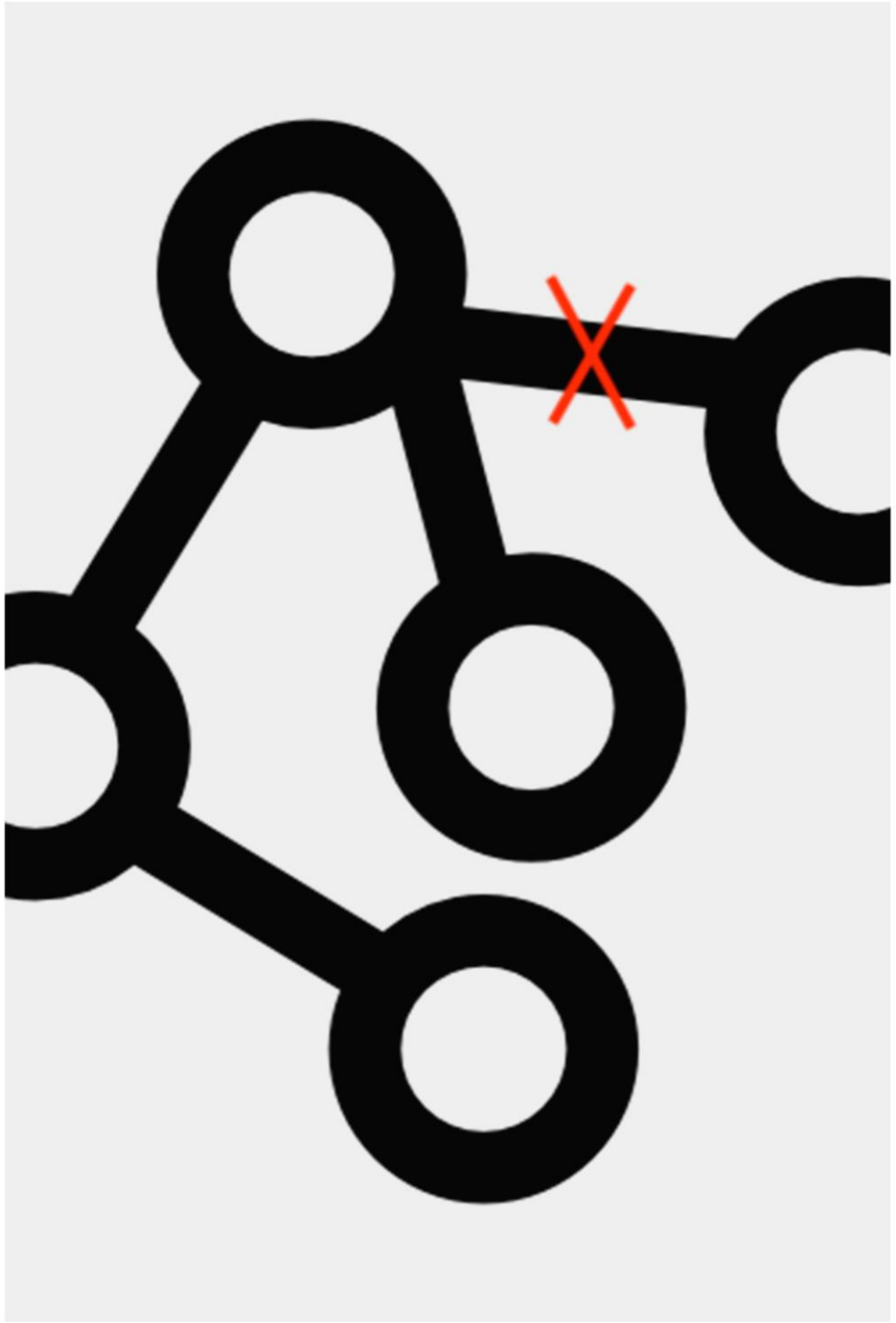
01. Introduction

How we got here?

- Endava Senior Test Consultant
- Decade spent in testing taking various roles in the team
- Part of the team who implemented a resilience test solution during modernisation program

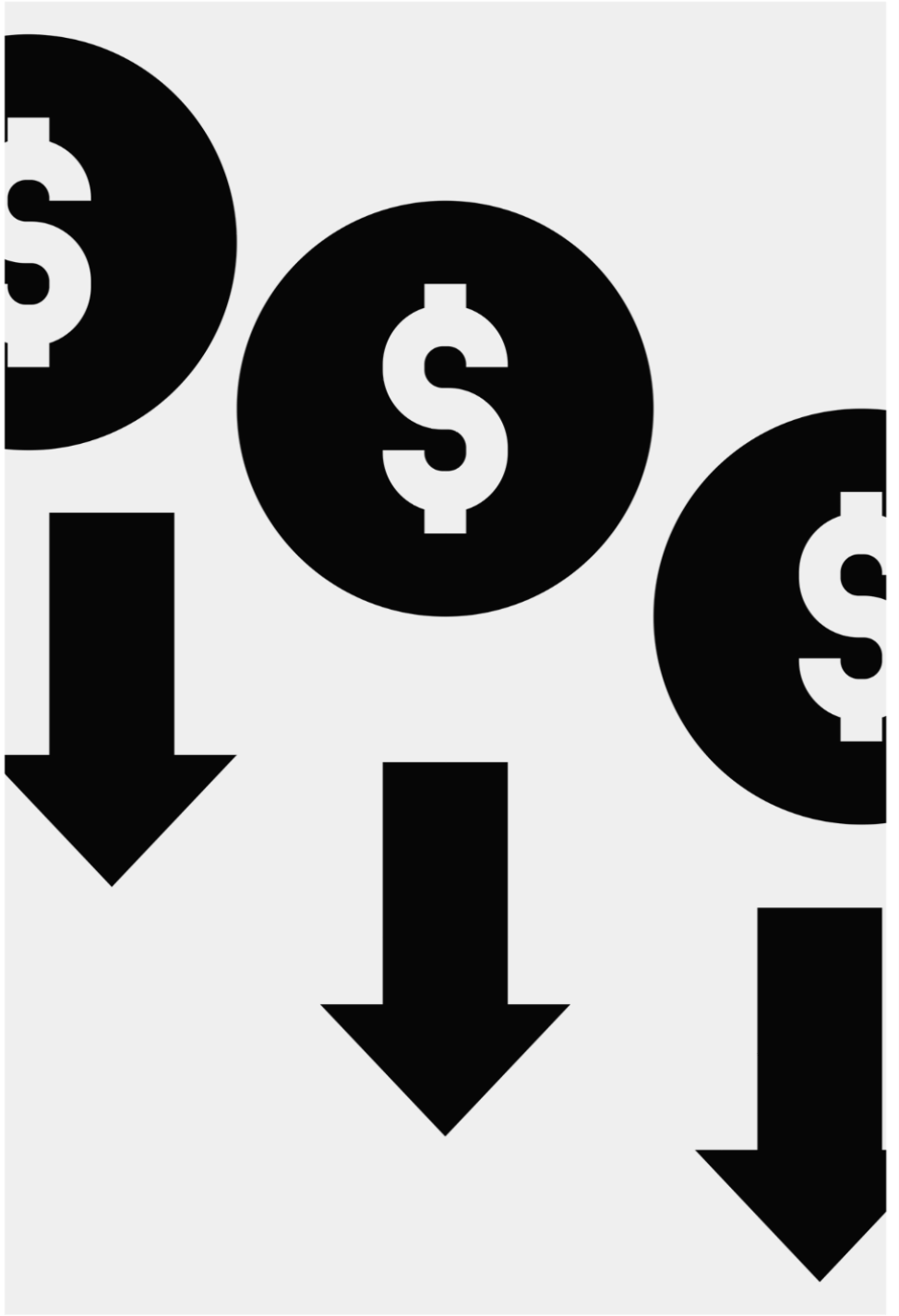
What is resilience testing?

- Non-Functional type of testing
- How system reacts to failures
- Many aspects to focus on
- Manual or automated
- Specific to a system under test



Why it is important?

- Switch to microservice architecture is adding layer of complexity to service communication
- Resilience issues can lead to financial, reputation, integrity loses and/or impact health and safety
- Findings can be used to improve disaster recovery plan
- Can help with optimizing resourse utilization



Focus points

Tools

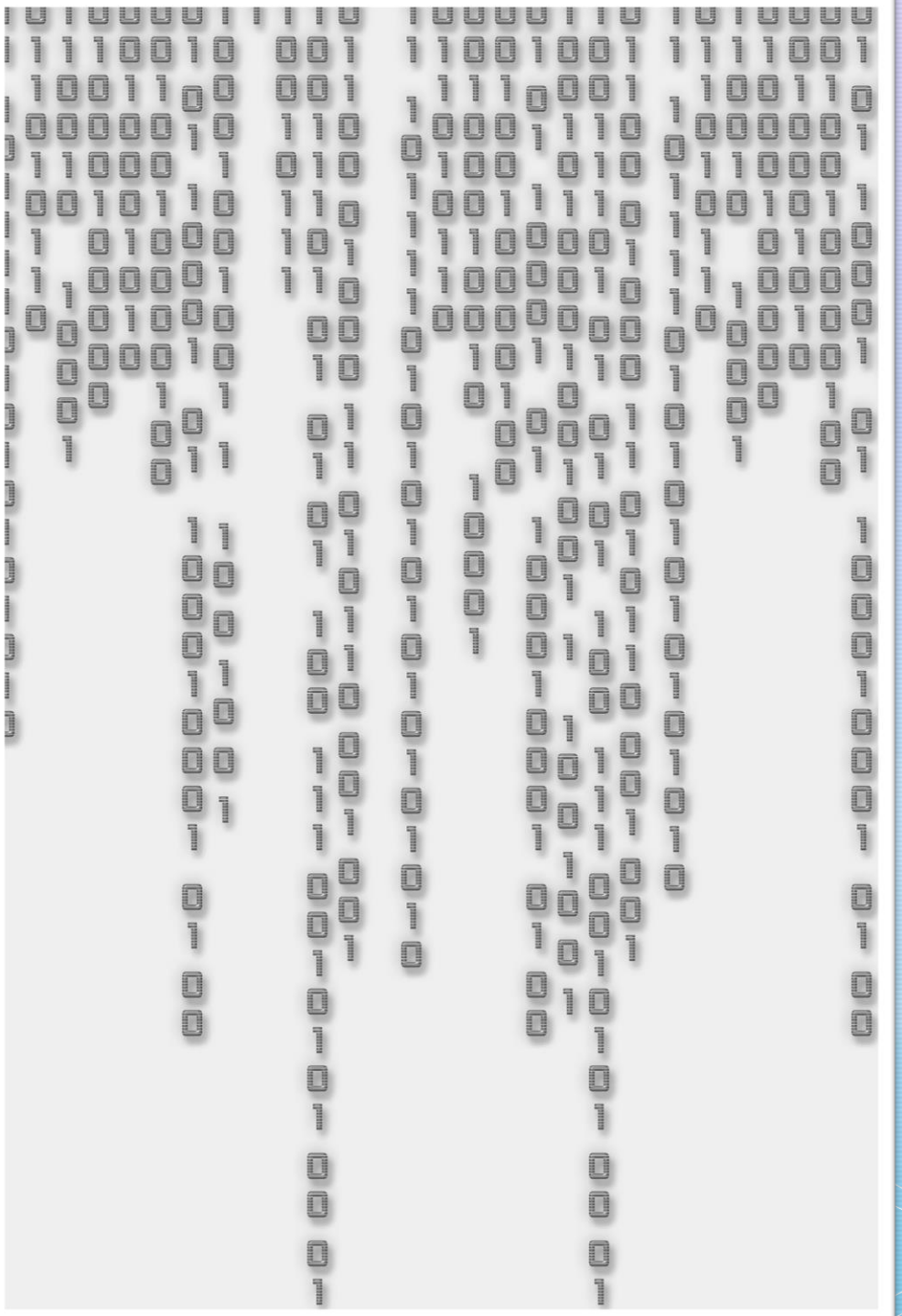
02. Implementation

Focus points
-> Service Discovery

- Health endpoints
- Status updates
- Service Mesh

Focus points -> Logs

- The Goldilocks principle
- Automated monitoring
- What, Who and When caused mailfunction



Focus points -> Connection Config

- Retry and timeout logic
- Failover and grace period logic
- Connection pool config

```
-- BudapestService
budapest.url=http://localhost:9092
budapest.url.context=/bankservice
budapest.http.connection.pool.size=50
budapest.http.read.timeout=80000
budapest.http.connection.timeout=5000
budapest.http.retry.grace.period=1000
budapest.http.gateway.max.connections=20000
budapest.http.max.route.connections=2000
budapest.http.ignore.ssl=true
budapest.http.failover.list=http://101.102.8.123:9092/
budapest.http.failover.list=http://localhost:9092
budapest.failover.tries=3


#-------------------------------------------------------------
#-------------------- Configuration for new HUSTEF Gateway
#-------------------------------------------------------------
#//http://<ip address>:<port>
hustef.rest.url=http://192.168.3.84:8080
hustef.rest.url.context=/process
hustef.rest.http.connection.pool.size=1000
hustef.rest.http.read.timeout=25000
hustef.rest.http.connection.timeout=2000
hustef.rest.http.retry.grace.period=1000
hustef.rest.http.gateway.max.connections=2500
hustef.rest.http.max.route.connections=60
hustef.rest.http.ignore.ssl=false
hustef.rest.failover.tries=3
hustef.rest.rest.http.failover.list=http://192.168.3.84:80

# Card Storage Service REST Template
hustef.http.connectionTimeout=5000
hustef.http.failoverList=http://localhost:9094
hustef.http.ignoreSSL=true
hustef.http.maxConnectionsPerRoute=300
hustef.http.maxTotalConnections=300
hustef.http.readTimeout=5000
hustef.http.retryGracePeriod=1000
hustef.http.tries=3
hustef.url=http://localhost:9094

```

Focus points
-> Fault scenarios

- Dropped connection, garbled responses
- Random data, Empty responses, Unexpected or missing fields
- Large payloads, Special characters…

```
POST /imposters HTTP/1.1
Host: localhost:38275
Accept: application/json
Content-Type: application/json

{
  "port": 4554,
  "protocol": "tcp",
  "mode": "text",
  "stubs": [
    {
      "responses": [
        {
          "fault": "CONNECTION_RESET_BY_PEER"
        }
      ]
    }
  ]
}
```

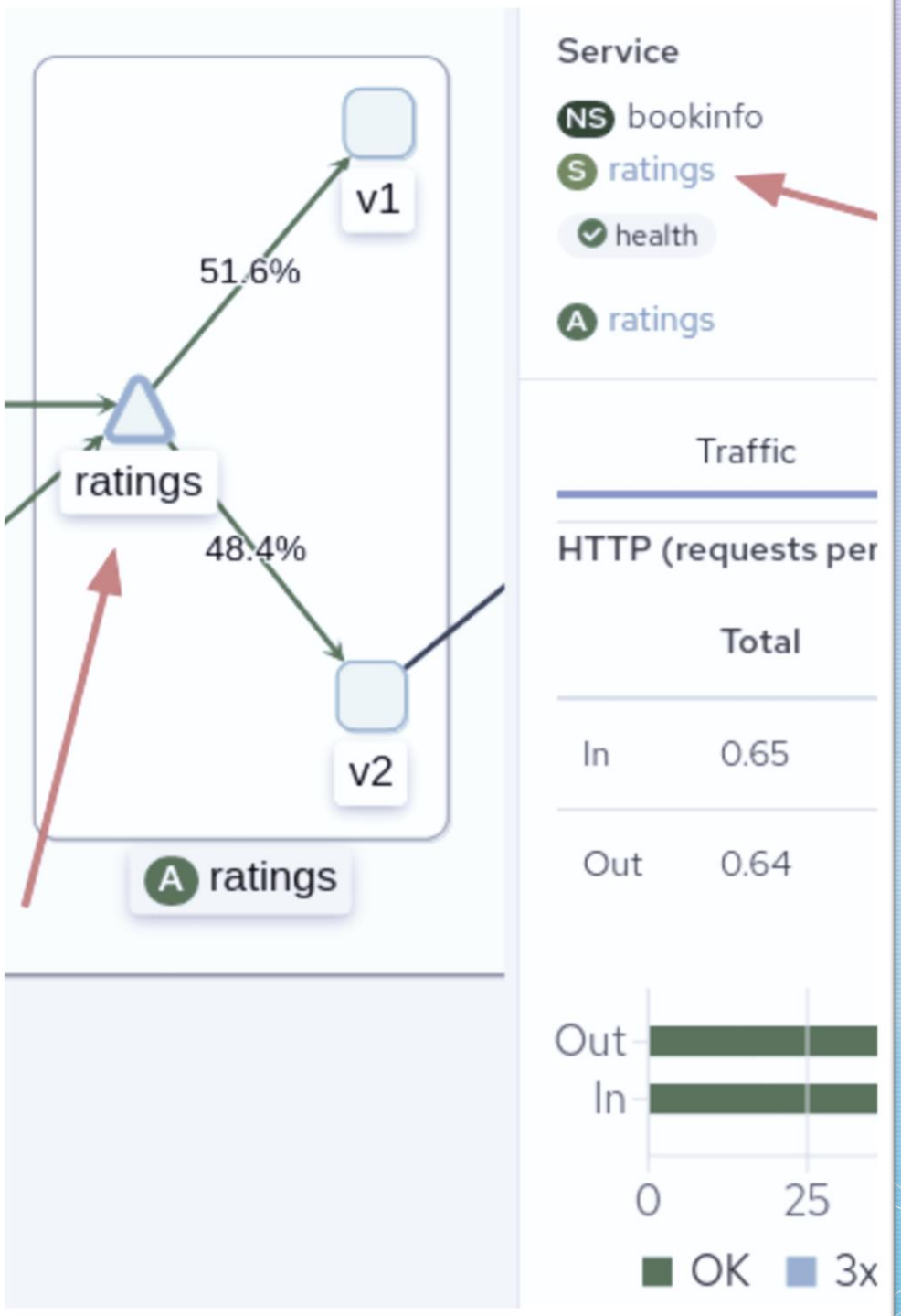
Random Data then Close

```
POST /imposters HTTP/1.1
Host: localhost:38275
Accept: application/json
Content-Type: application/json

{
  "port": 4554,
  "protocol": "tcp",
  "mode": "text",
  "stubs": [
    {
      "responses": [
        {
          "fault": "RANDOM_DATA_THEN_CLOSE"
        }
      ]
    }
  ]
```

Focus points -> Tools

- Standard Automation frameworks and IDEs
- Mocks (Mountebank, Wiremock…)
- Service Mesh (Consul, Istio, Kiali...)



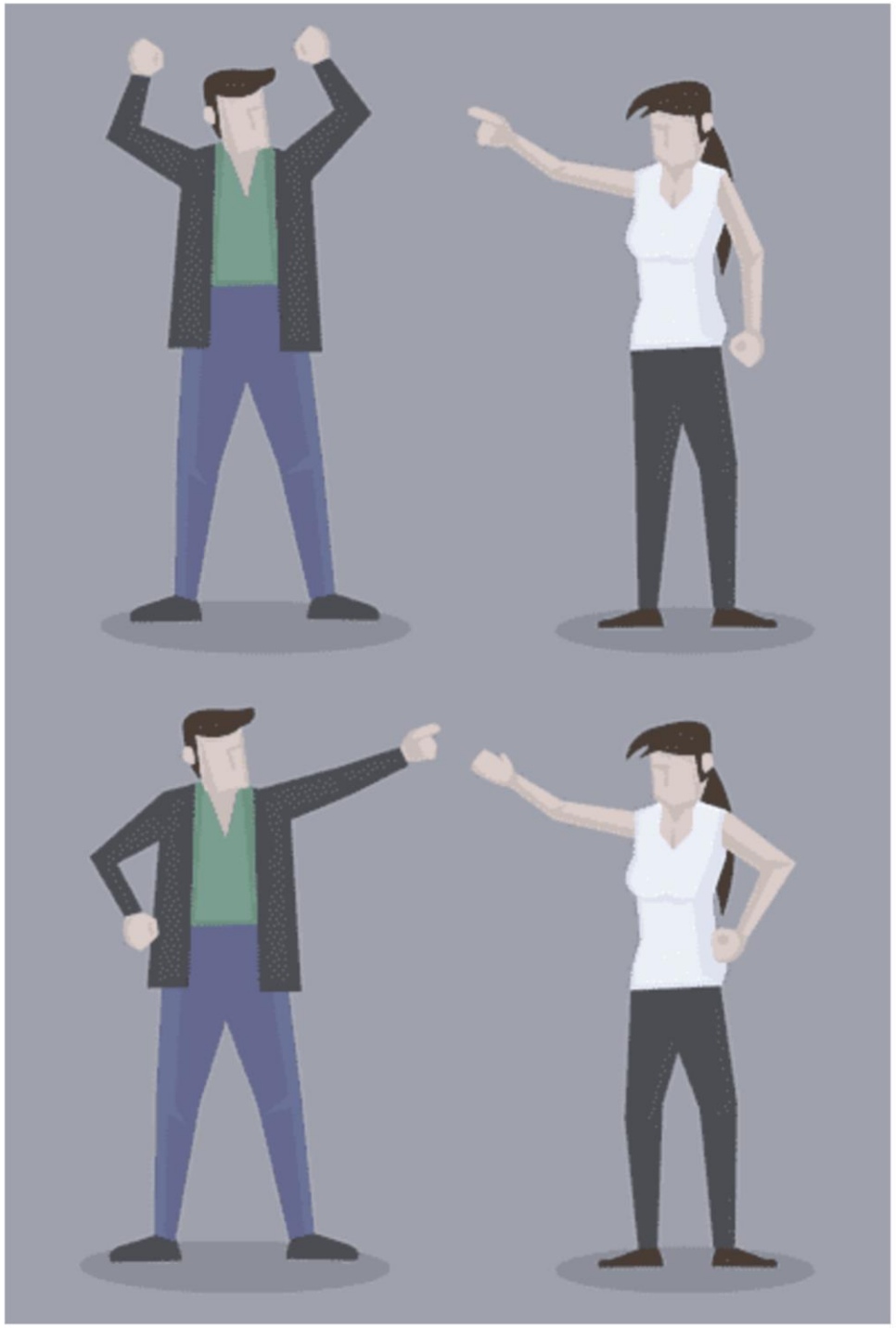
Team communication

Learning about the system

03. Challenges

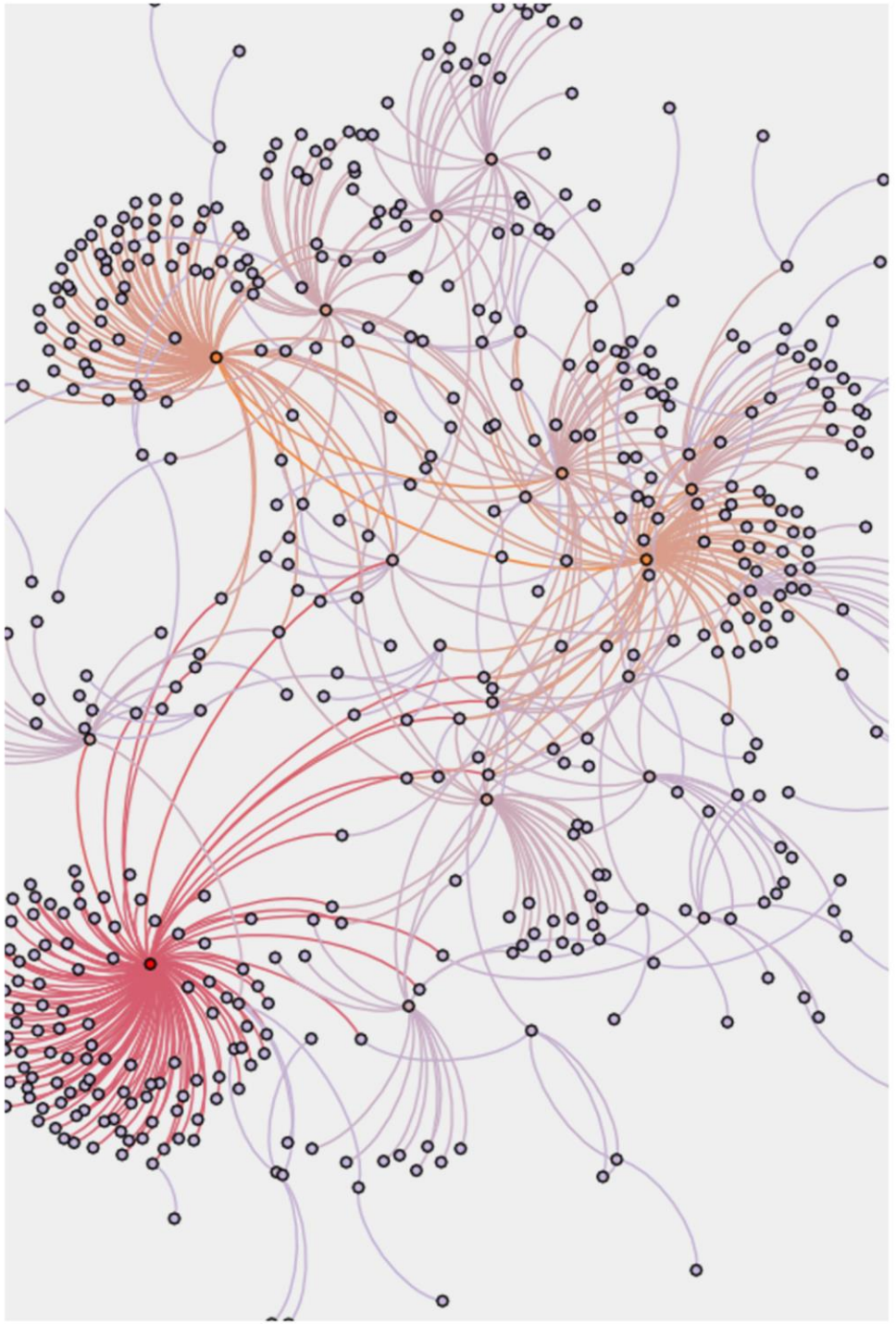
Team Communication

- Allocating enough time
- Service standardization
- Team resistance



Learning about a system

- 70% of time for existing system
- Proxy stubs
- Kiali



Include resilience early

04. Conclusion

Include resilience early

- Think about health endpoints and logs during design phase
- Plan testing connection aspects as soon as service is introduced
- Ask for standardization across services

???

05. Questions

Thank you!!