CHAOS ENGINEERING – PAST, PRESENT AND FUTURE

Vilas Veeraraghavan

Director of Engineering, Walmart

msup

TOP100Summit
全球软件案例研究峰会
时间：11月15~17日
地点：北京国际会议中心
100个年度最值得学习案例

MPD工作坊（深圳站）
时间：9月21~22日
地点：深圳博林圣海伦酒店
20个3小时大时段沙盘课程

MPD工作坊（北京站）
时间：7月06~07日
地点：北京国家会议中心
20个3小时大时段沙盘课程

MPD工作坊（上海站）
时间：10月26~27日
地点：上海
20个3小时大时段沙盘课程

THE JOURNEY

- Past (2010 - 2015)
  - Origins
  - Lessons learned
  - Tools
- Present (2016-2019)
  - Increasing adoption
  - Continued improvements and innovation
  - Mainstream tools
- Future (2019 -)
  - Automation
  - Organizational status

PART ONE - PAST

PIVOT TO CLOUD

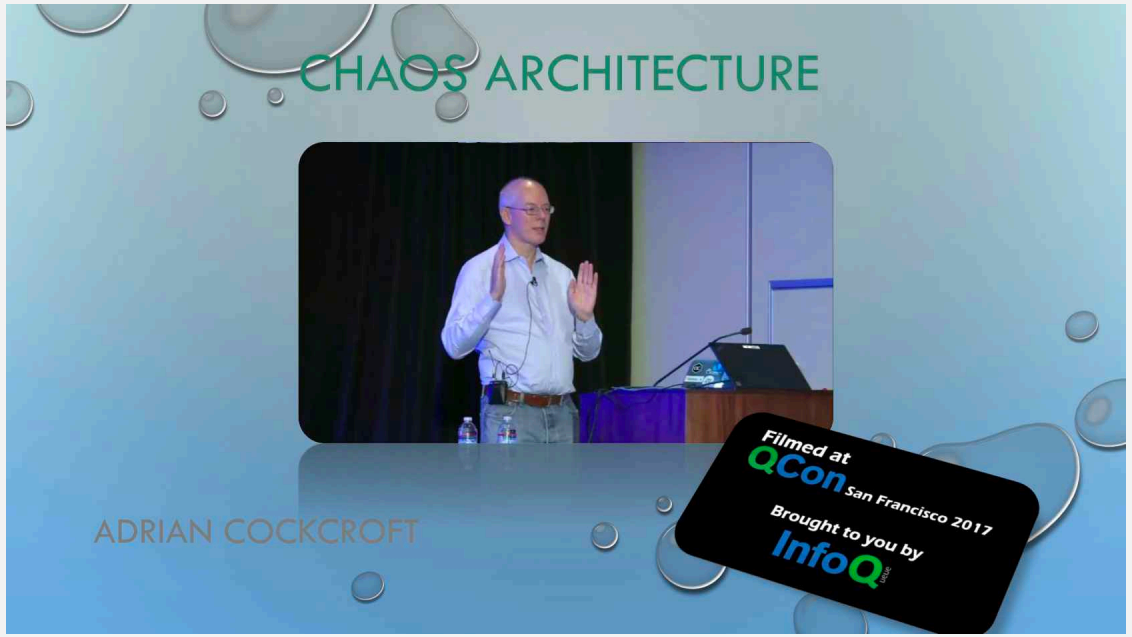
- Datacenter dependencies
- Emergence of cloud services
- Breaking up monoliths into microservices

PIVOT TO CLOUD

- Unreliability – Issues with the cloud providers
  - Multiple incidents with AWS
- How can we protect applications against outages?

REFERENCE

https://www.infoq.com/presentations/chaos-architecture-mindset



TOOLS

https://github.com/Netflix/chaosmonkey

HOWEVER

http://www.quickmeme.com/meme/3oh4hu

DETERMINISTIC FAILURES

OBSERVABILITY

- Logging alone is not enough
- Ability to differentiate between healthy and unhealthy behavior
- Alerts set up
- Make sure your on-call knows what to do for specific problems

LESSONS LEARNED

- Resiliency cannot be achieved by injecting chaos without control
- Running a chaos exercise is NOT an individual effort.
- Observability of metrics is key to get value out of exercises

CLOUD ARCHITECTURE

Users/Frontend

Internet

FIREWALL

Load Balancer

Application server

Application server

Region A

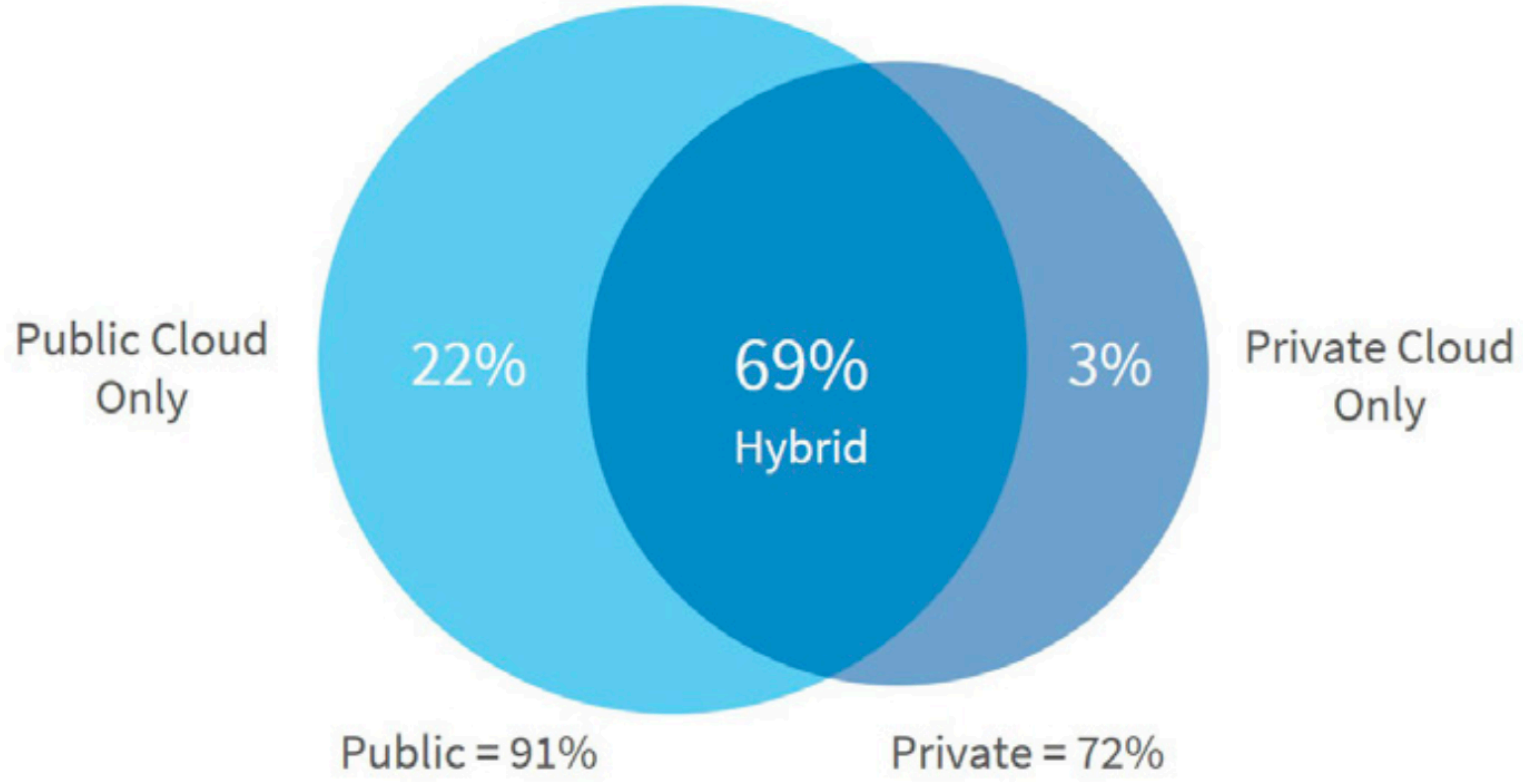
Region B

Database

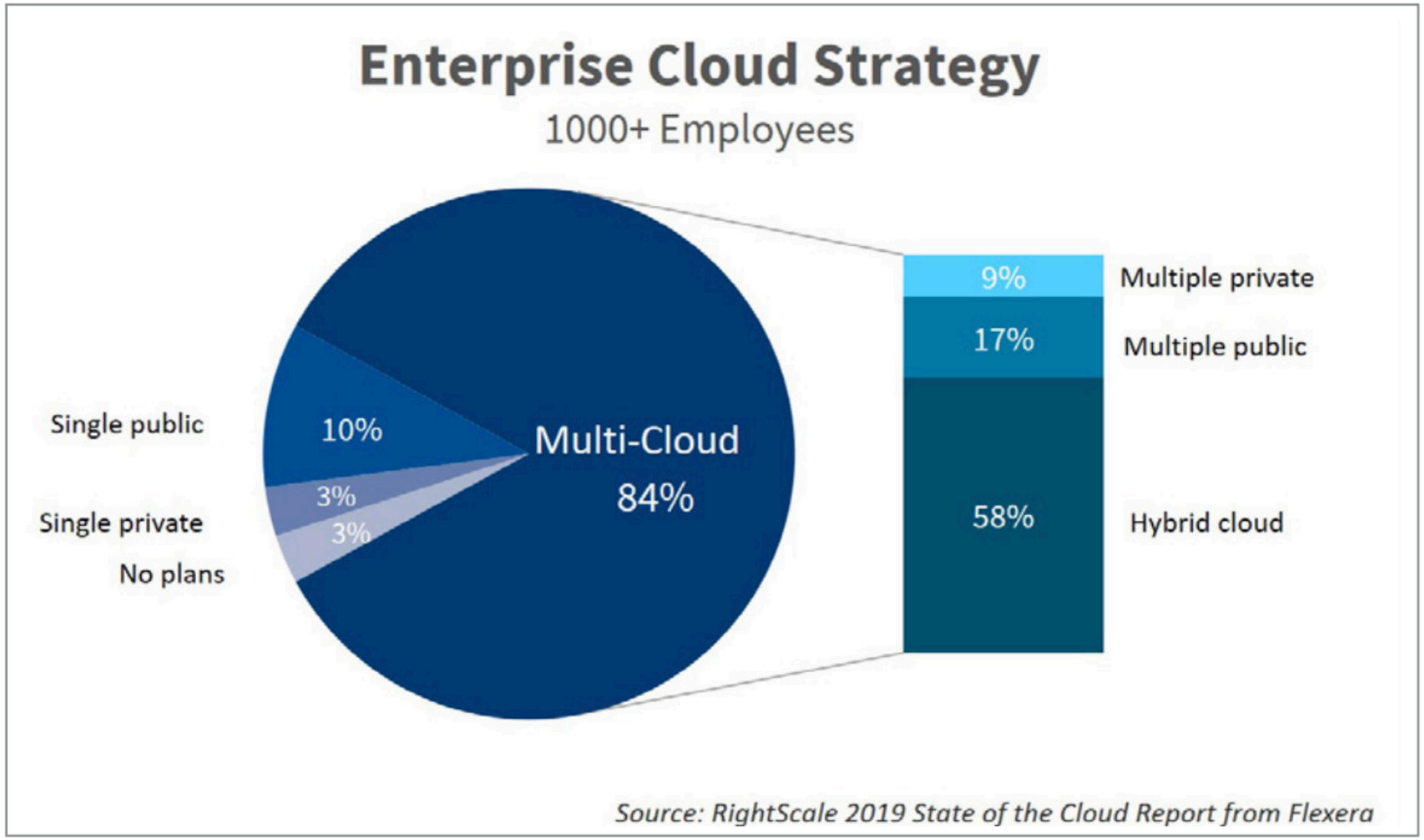
Database

PART TWO - PRESENT

94% of Respondents Using Cloud



Source: RightScale 2019 State of the Cloud Report from Flexera


Enterprise Cloud Strategy
1000+ Employees
Single public
10%
Single private
3%
No plans
3%
Multi-Cloud
84%
9%
Multiple private
17%
Multiple public
58%
Hybrid cloud
Source: RightScale 2019 State of the Cloud Report from Flexera

MOTIVATIONS

- Reliability is no longer a function of redundancy and over-scaled hardware.
  - Specifically, to exist in a hybrid cloud environment, we have to acknowledge that cloud providers are an external dependency which are reliability risks.
- Customers expect more and ‘scheduled downtime’ is no longer an acceptable term.
  - A user performing transactions (search, add to cart, payment) should not perceive a loss of functionality due to systemic failure.

MOTIVATIONS

- Users can lose trust on brand due to a single bad experience
- Loss could be temporary OR lifetime
- Resiliency is not a local goal – it is a global goal

GOAL

To maintain an application ecosystem where failures in infrastructure and dependencies cause minimal disruption to the end user experience

WHAT IS CHAOS ENGINEERING?

- It is NOT to be confused with Integration or Performance testing
- Integration testing is ensuring that functional requirements and contracts with teams have been met
- Performance vs resiliency – what is the difference?
  - You can be performant but not resilient
  - And vice-versa

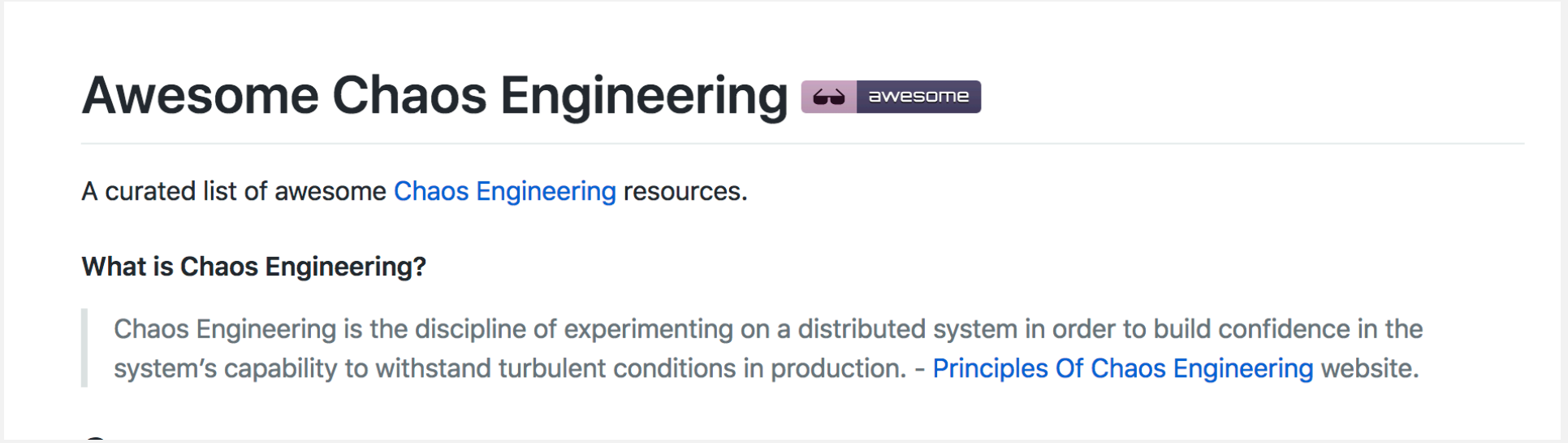
https://principlesofchaos.org/

PRESENT

- Chaos engineering has gone mainstream
- Big companies use it now – Walmart, Nike, Target
- You can use it too!

CREATING RESILIENT SYSTEMS USING CHAOS ENGINEERING

TOOLS



Awesome Chaos Engineering

A curated list of awesome Chaos Engineering resources.

What is Chaos Engineering?

> Chaos Engineering is the discipline of experimenting on a distributed system in order to build confidence in the system's capability to withstand turbulent conditions in production. - Principles Of Chaos Engineering website.

https://github.com/dastergon/awesome-chaos-engineering

TECHNIQUES

- Engineering organization needs to be motivated
  - Incentivize teams to conduct scientific experiments and FAIL!
- SRE team charter needs to include chaos
- Establish a core practitioners group inside the company
  - These are the key tech leads from all the technology groups

TEAM

- Everyone has a hand in the success
- Cant hire everyone – make the right choices – hire senior engineers
- Little bit of everything – different people have different skills
- Counter-intuitive - Incentivize preventing outages by causing them

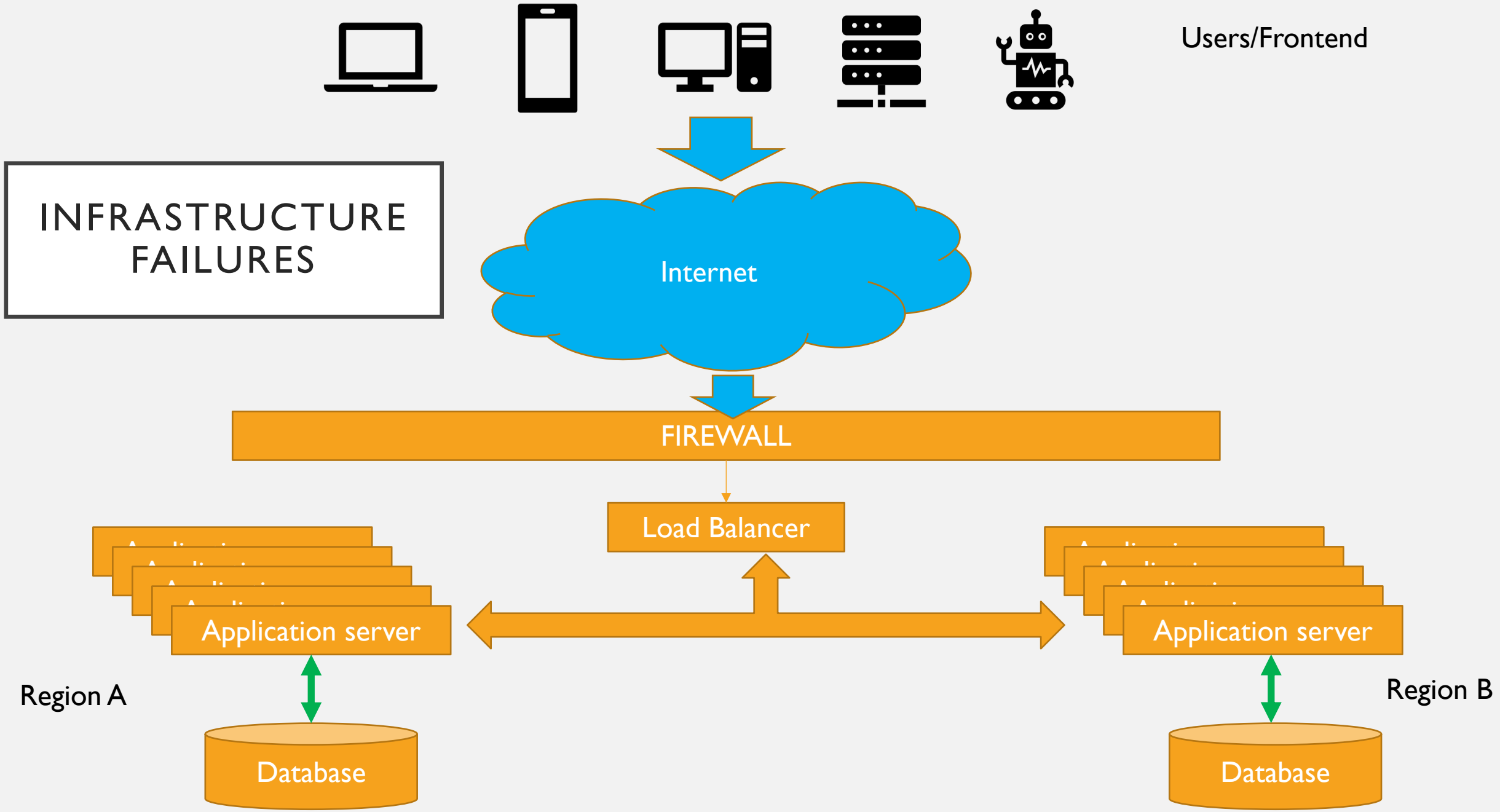
CONVINCING MANAGEMENT

- “Resiliency” is the end goal NOT chaos
- Use outages as the way to motivate management
- Do your homework!! – calculate support costs

KEY TESTS

- Infrastructure issues – failures, glitches, faulty maintenance policies
- Dependency failures – changing versions of APIs, changing SLAs
- Deployment issues – is the app even deployed right?

INFRASTRUCTURE FAILURES



INFRASTRUCTURE FAILURES

Users/Frontend

Internet

FIREWALL

Load Balancer

Application server

Application server

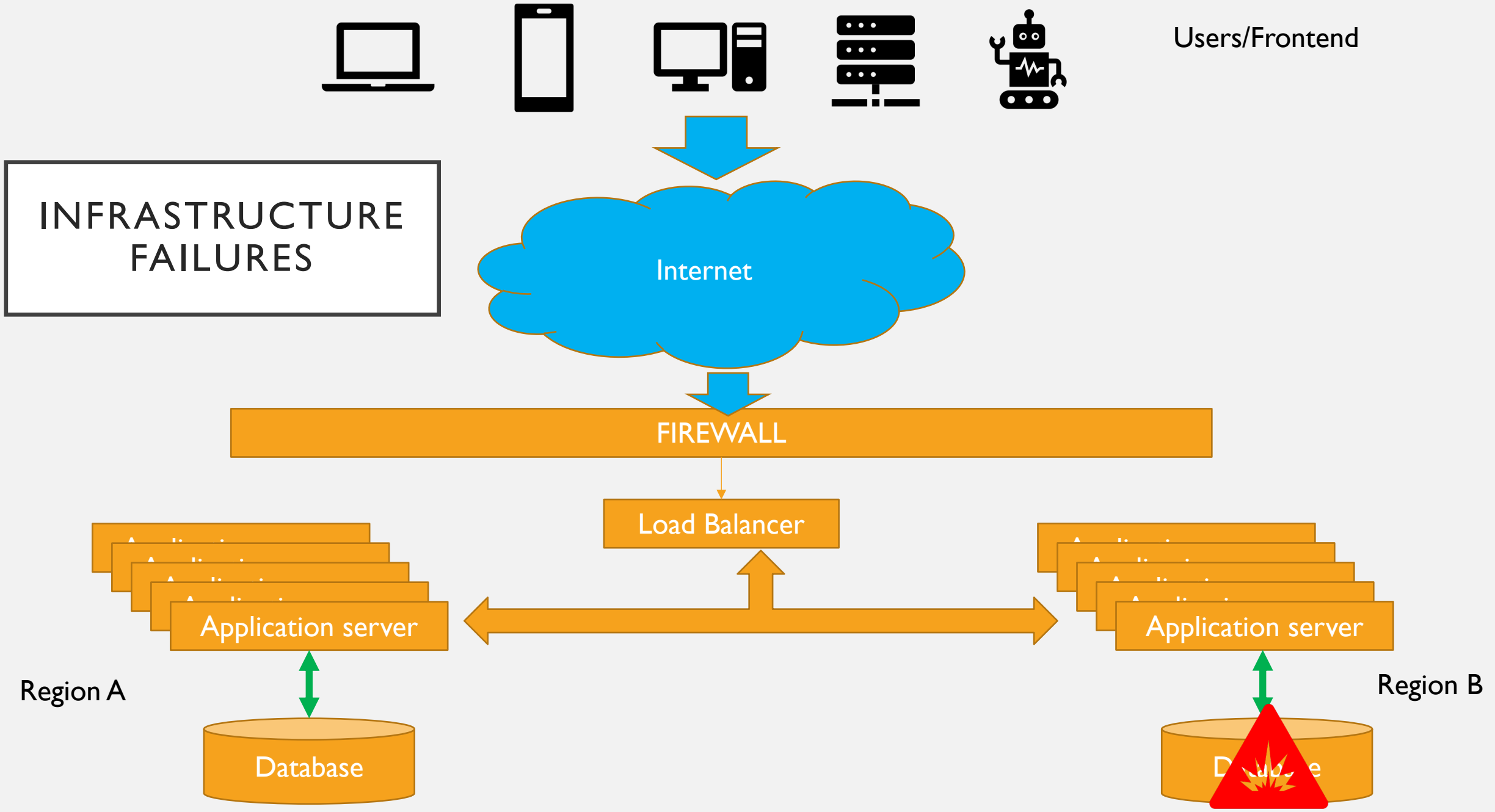
Region A

Region B

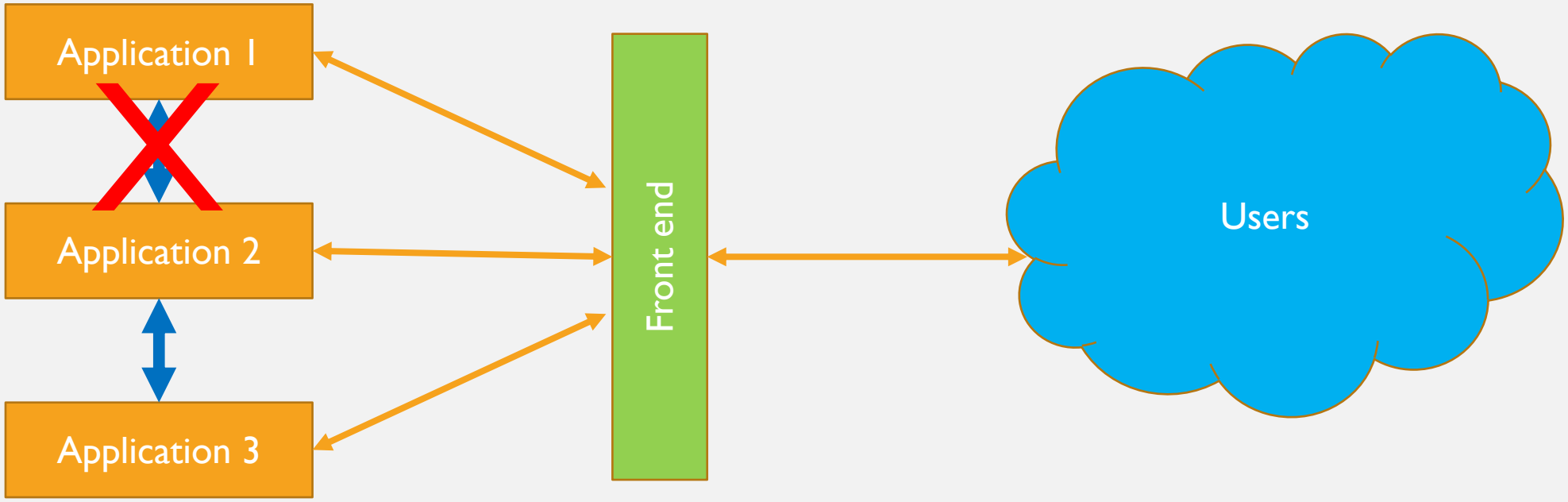
Database

Database

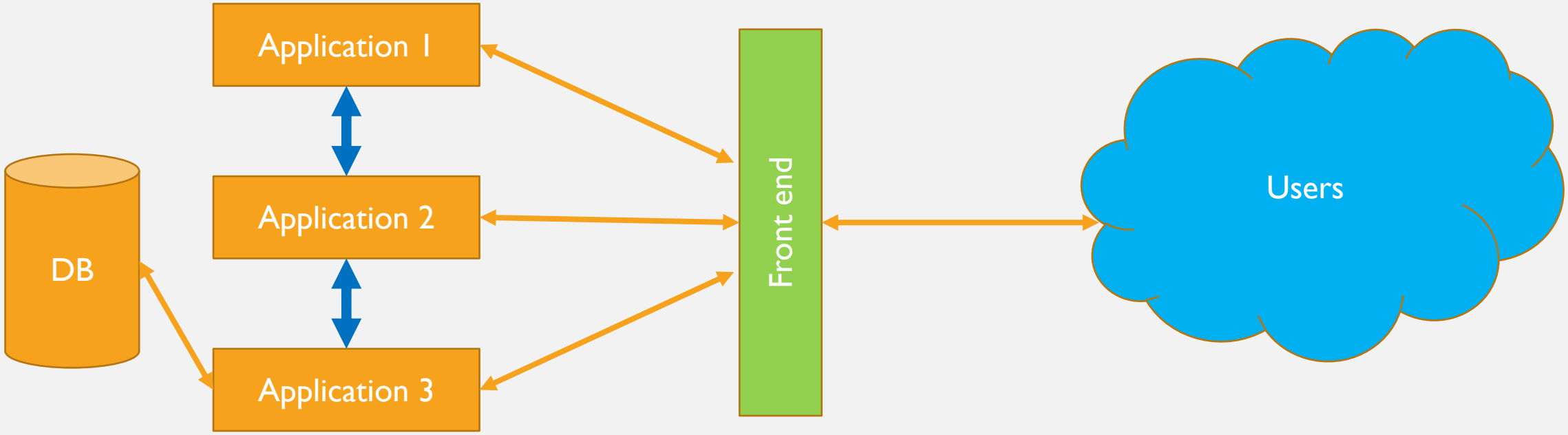
INFRASTRUCTURE FAILURES



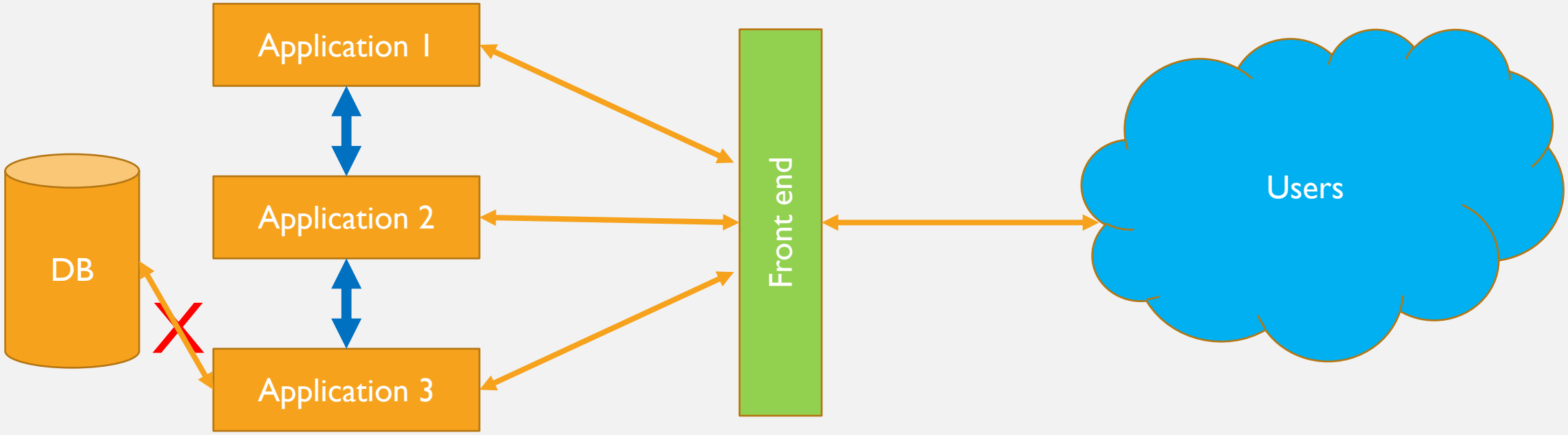
DEPENDENCY FAILURES



DEPENDENCY FAILURES



DEPENDENCY FAILURES



PREREQUISITES

- Create Disaster Recovery (DR) failover playbook
- Define critical dependencies
- Compose playbook for critical dependency failures
- Define non-critical dependencies
- Define thresholds at which non-critical dependency failures will impact system

EXPERIMENT

1. Identify what you intend to fail
2. Before : Write down hypothesis of system behavior based on known assumptions
3. Inform key stakeholders
4. Run test
5. After : verify if hypothesis holds
6. If hypothesis does not hold, analyze and fix. If it holds, hypothesis is validated

EXPERIMENT

1. Identify what you intend to fail
2. Before : Write down hypothesis of system behavior based on known assumptions
3. Inform key stakeholders
4. Run test
5. After : verify if hypothesis holds
6. If hypothesis does not hold, analyze and fix. If it holds, hypothesis is validated

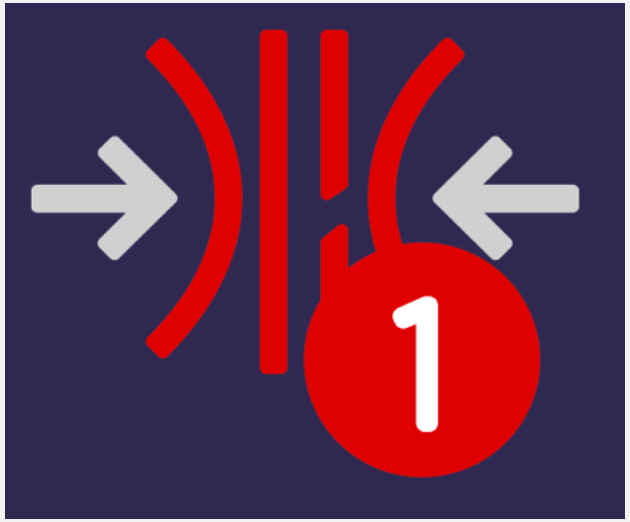
Repeat!!

LEVELS

resiliency

LEVEL 1

- **All** of the pre-requisites stored in a single well-defined place
- Agreement on playbooks to be used by Devs, Testers, Operations, Stakeholders
- Manual exercise that validates the DR failover playbook



LEVEL 2

- **All** of level 1 requirements, plus
- Run a failure test for critical dependencies in a non-prod environment
- Publish test results to team, stakeholders
- Manual tests are acceptable

LEVEL 3

- **All** of level 2 requirements, plus
- Run tests regularly on a cadence (at least once every 4–5 weeks)
- Publish results to dashboards to track resiliency over time
- Run at least one resiliency exercise (failure injection) in production environment

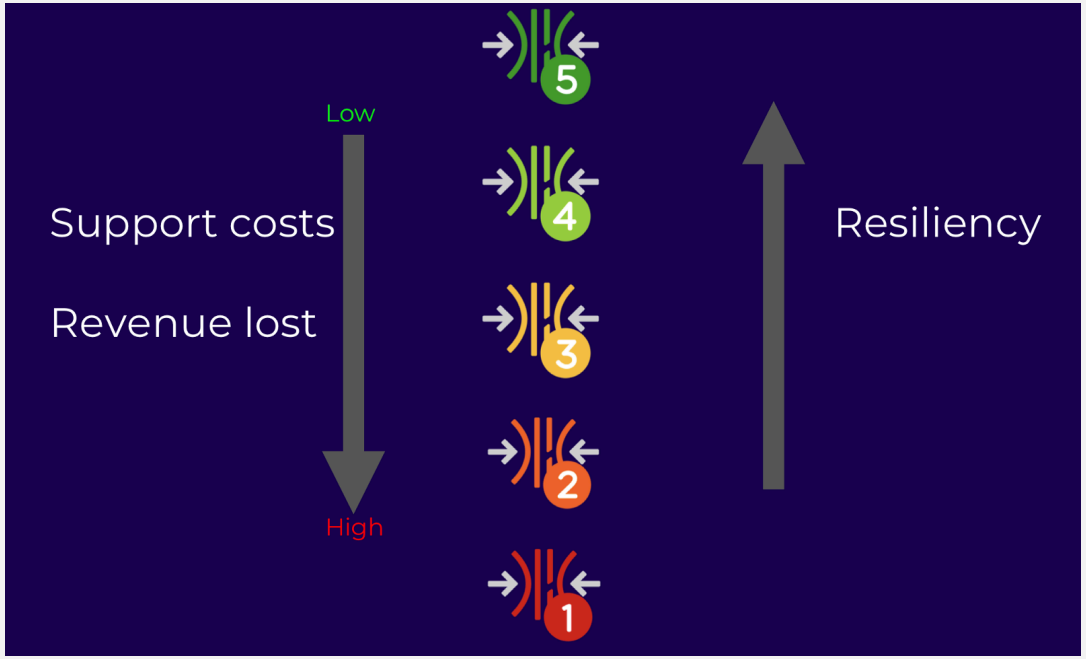
LEVEL 4

- **All** of level 3 requirements, plus
- Automated resiliency testing in non-prod environment
- Semi-automated DR failover scripts (minimal human supervision required)

LEVEL 5

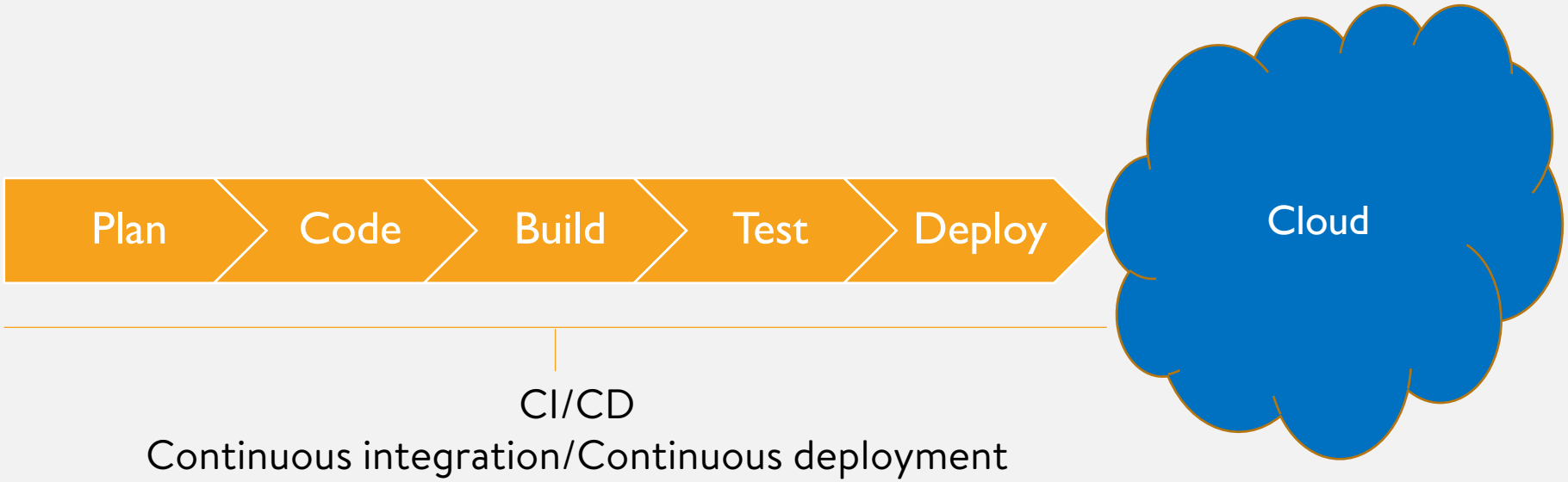
- **All** of level 4 requirements, plus
- Automated resiliency testing fully integrated into CI/CD environment
- Resiliency failure results in build failure
- Automated resiliency testing and DR failover testing enabled in production environment

SUPPORT COSTS



PART THREE - FUTURE

CI/CD


Plan
Code
Build
Test
Deploy
Cloud
CI/CD
Continuous integration/Continuous deployment

CI/CD

- Make resiliency testing a part of the continuous integration cycle
- Continuous delivery pipelines can have checks on resiliency metrics

https://concord.walmartlabs.com/

EXAMPLE

Blackhole

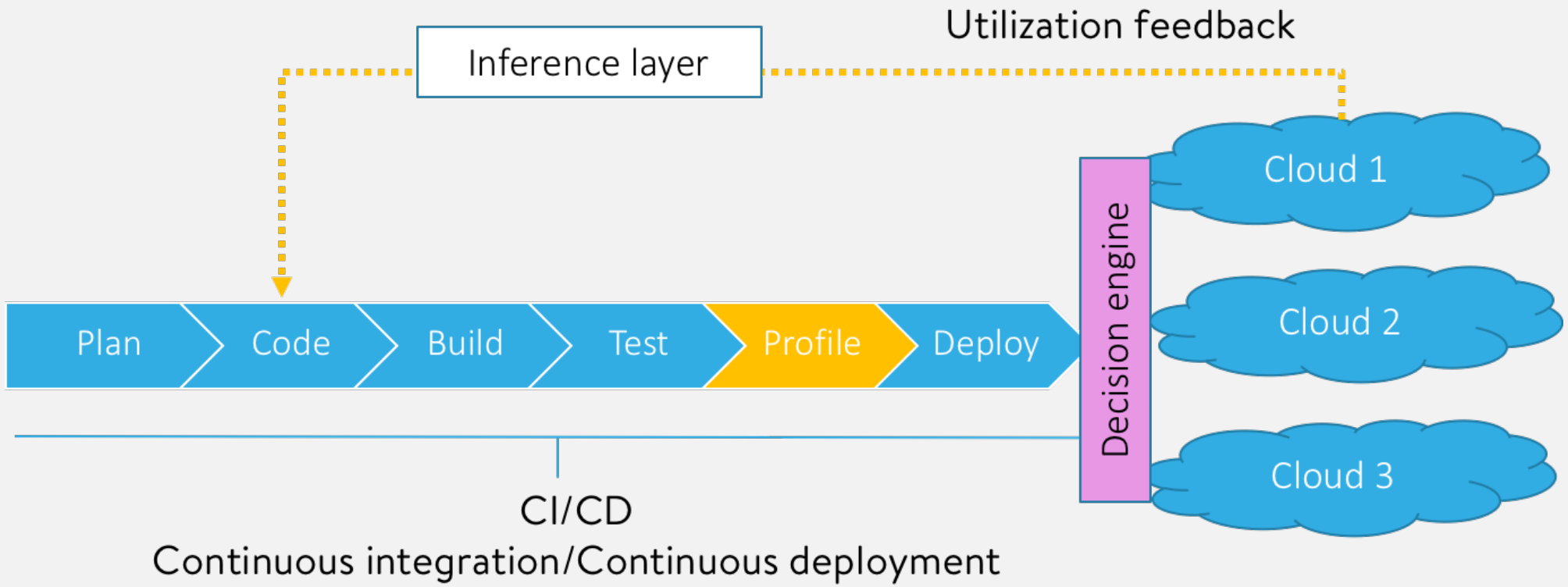
The `blackhole` action of the `gremlin` task can be used drop all matching network traffic. The following parameters are needed in addition to the general parameters:

- `ipAddresses`: Required - Impact traffic to these IP addresses
- `device`: Impact traffic over this network interface
- `hostnames`: Only impact traffic to these hostnames. Whitelist a host with a leading `^`
- `egressPorts`: Only impact egress traffic to these destination ports. Ranges work too: `8080-8085`
- `ingressPorts`: Only impact ingress traffic on these incoming ports. Ranges work too: `8080-8085`
- `protocol`: Only impact traffic using this IP protocol. Allowed values are TCP, UDP, ICMP. Defaults to all protocols

```yaml
- task: gremlin
  in:
    action: blackhole
    apiKey: myApiKey
    length: 15
    ipAddresses: "ipAddress1, ipAddress2"
    device: "myDevice"
    hostnames: "host1.com, host2.com"
    egressPorts: "egPort1, egPort2"
    ingressPorts: "ingPort1, ingPort2"
    protocol: UDP
    targetType: Exact
    targetList: ["client1", "client2"]
```

https://concord.walmartlabs.com/docs/plugins/gremlin.html

CI/CD – THE REAL PICTURE



TOOLS

- Gremlin, chaos toolkit, chaosblade
- Kubernetes based deployments also need chaos – use powerful seal

COMPLETE AUTOMATION

REFERENCES

- https://medium.com/netflix-techblog/from-chaos-to-control-testing-the-resiliency-of-netflixs-content-discovery-platform-ce5566aef0a4
- https://medium.com/walmartlabs/charting-a-path-to-software-resiliency-38148d956f4a
- https://www.youtube.com/watch?v=4Gy_5EQMrB4

THANK YOU

QUESTIONS??