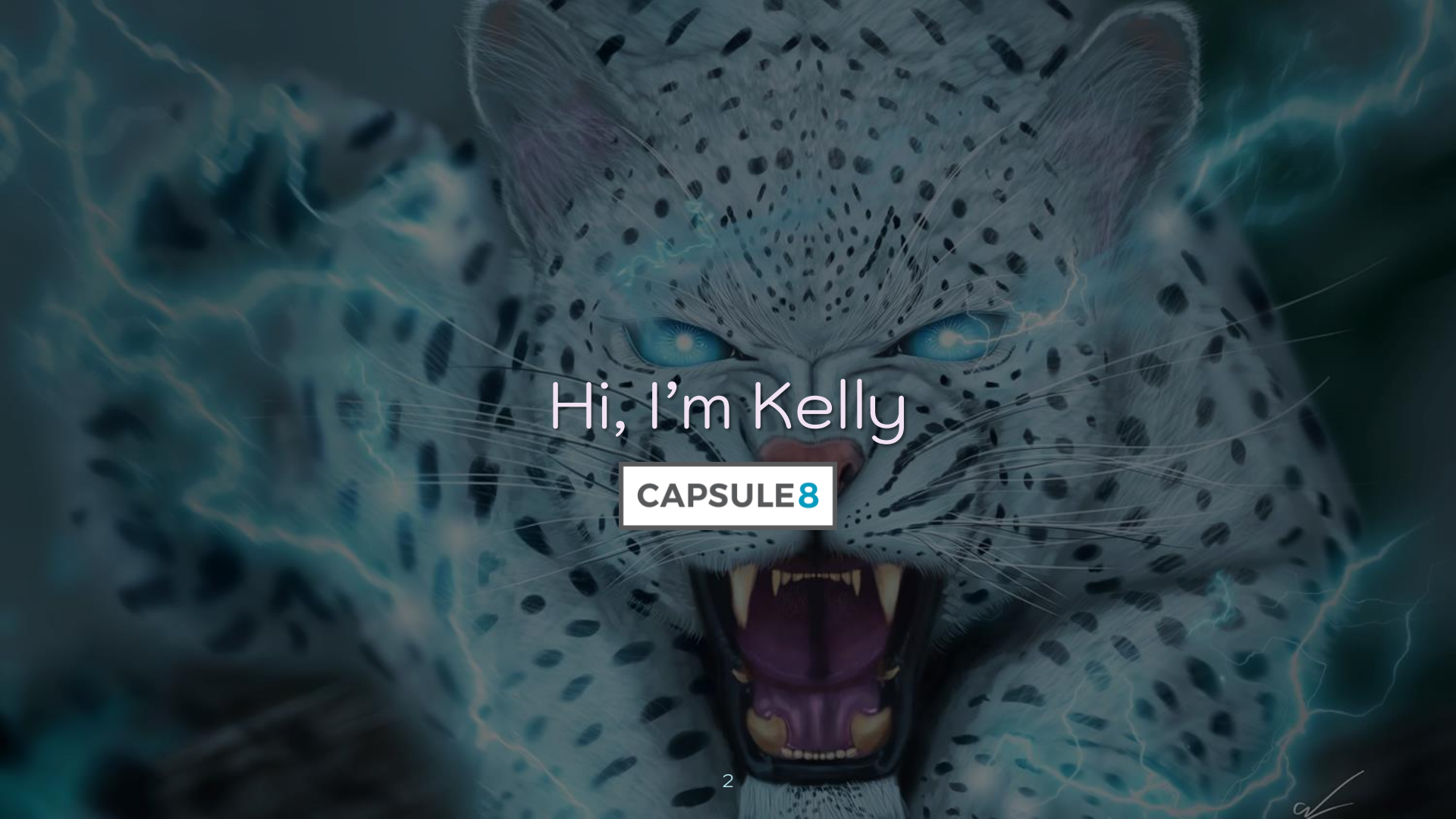
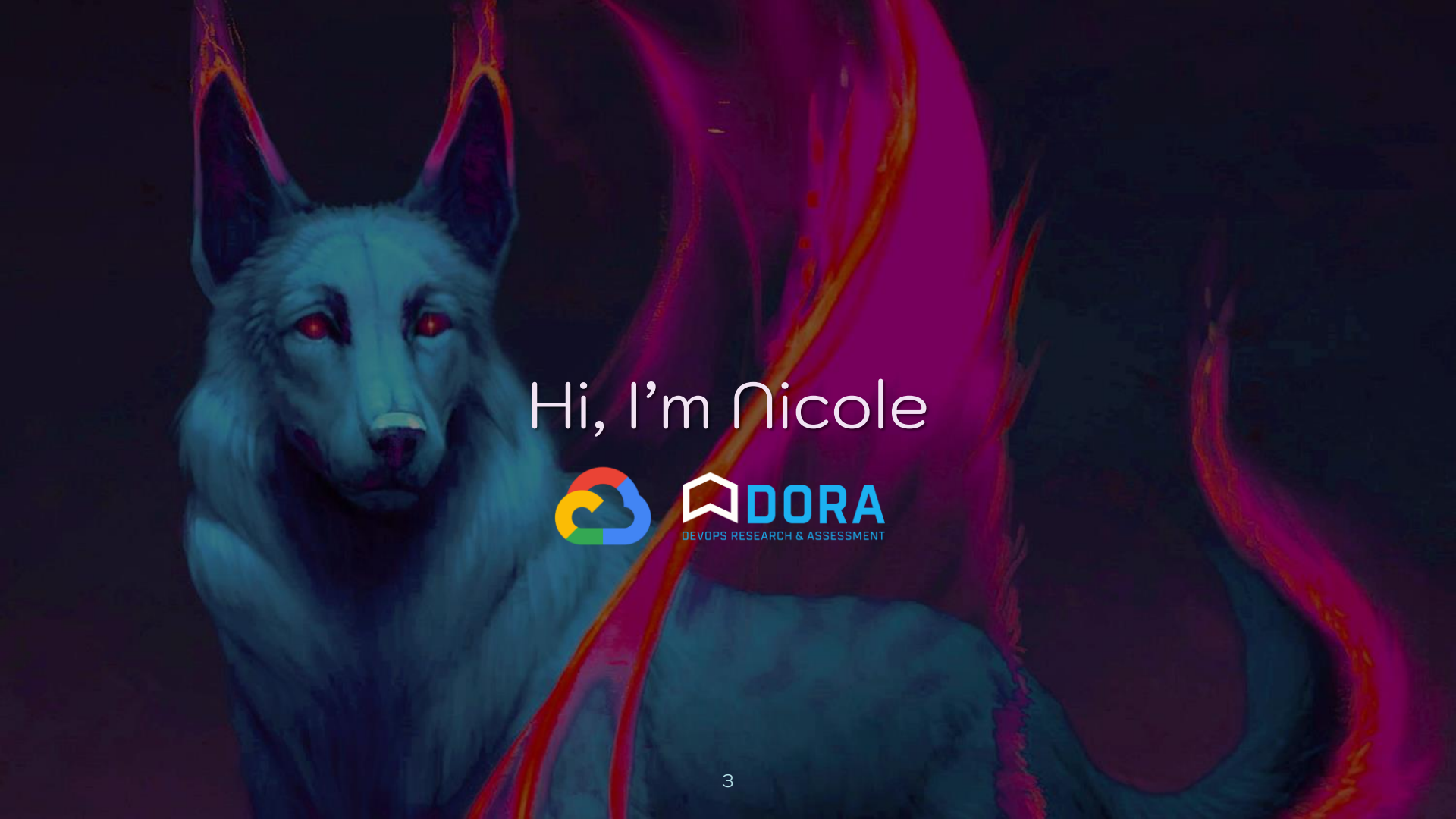
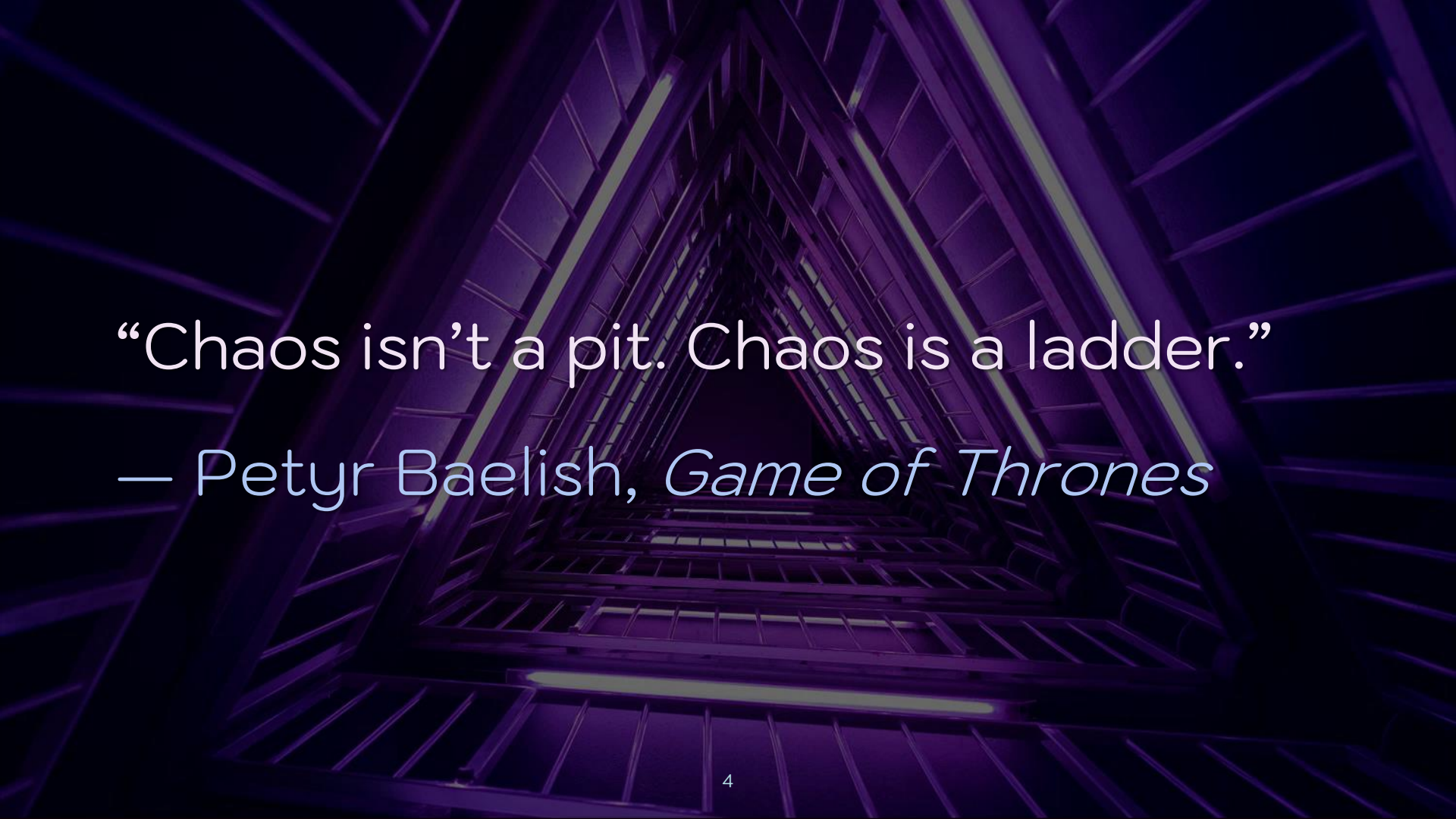

Controlled Chaos
The Inevitable Marriage of DevOps & Security
Kelly Shortridge (@swagitda_)
Dr. Nicole Forsgren (@nicolefv)
Black Hat USA 2019

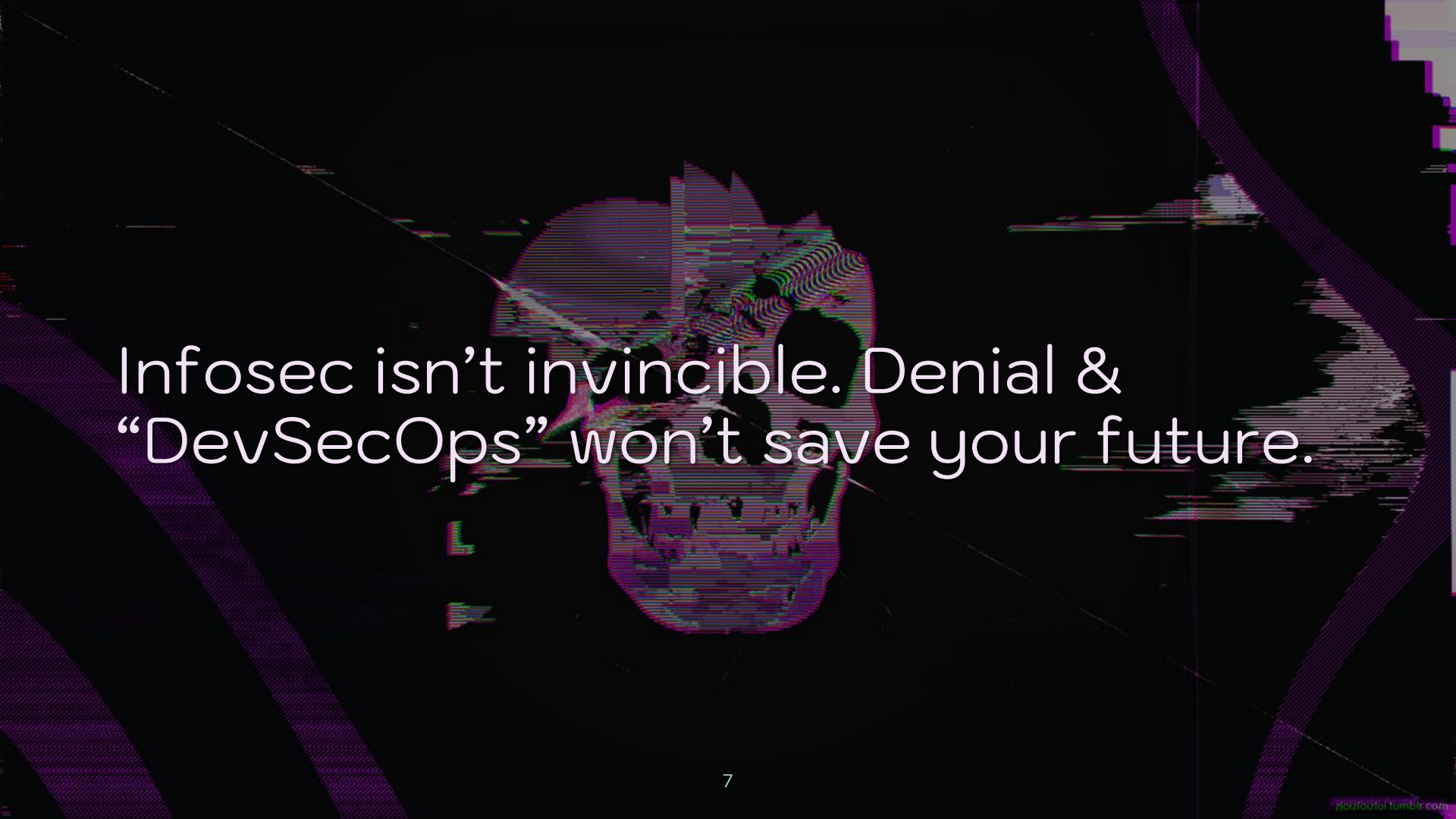

Hi, I'm Kelly
CAPSULE8

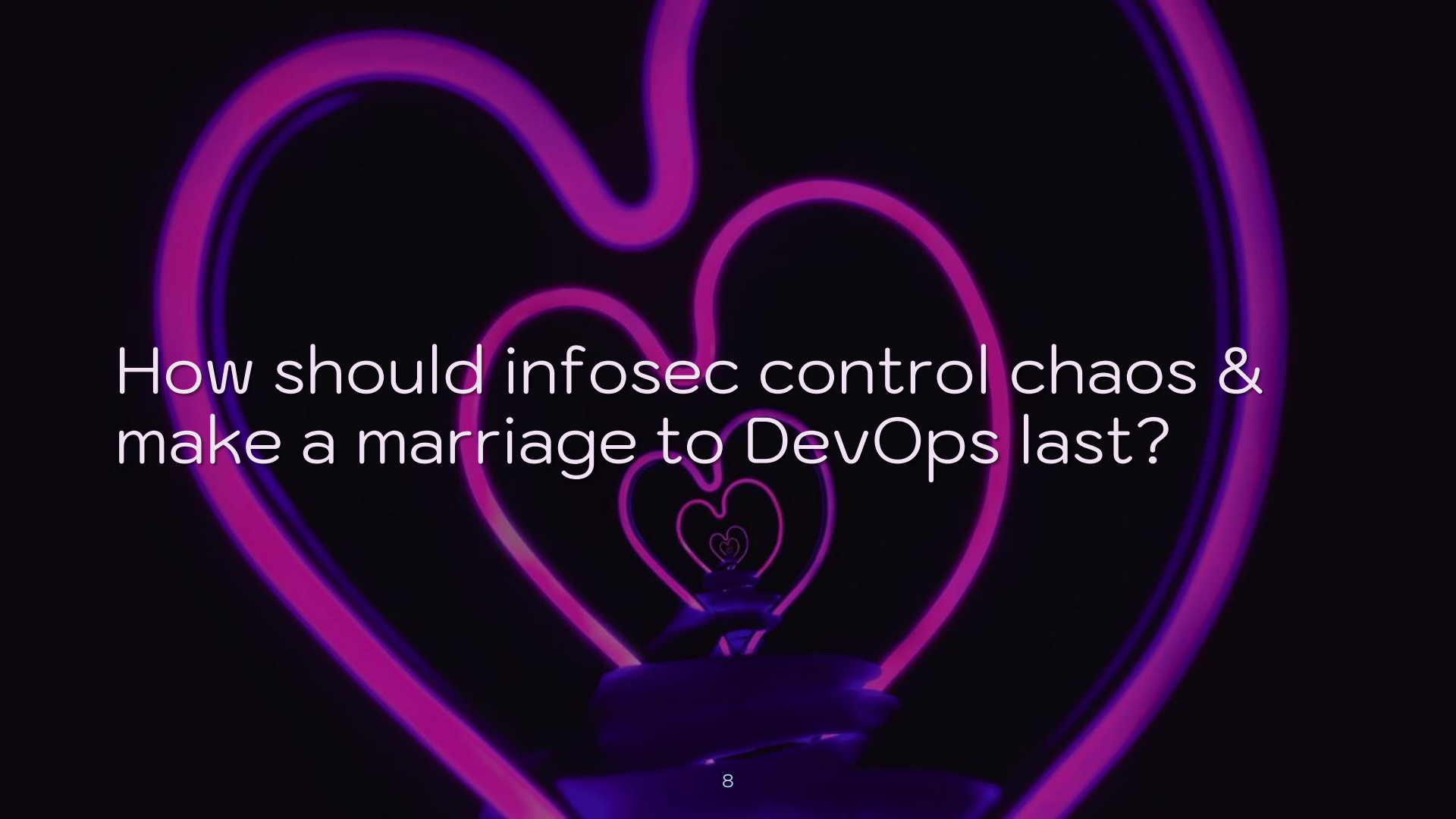

Hi, I'm Nicole
DORA
DEVOPS RESEARCH & ASSESSMENT


“Chaos isn’t a pit. Chaos is a ladder.”
— Petyr Baelish, *Game of Thrones*


Software is eating the world.
DevOps drives its devouring.

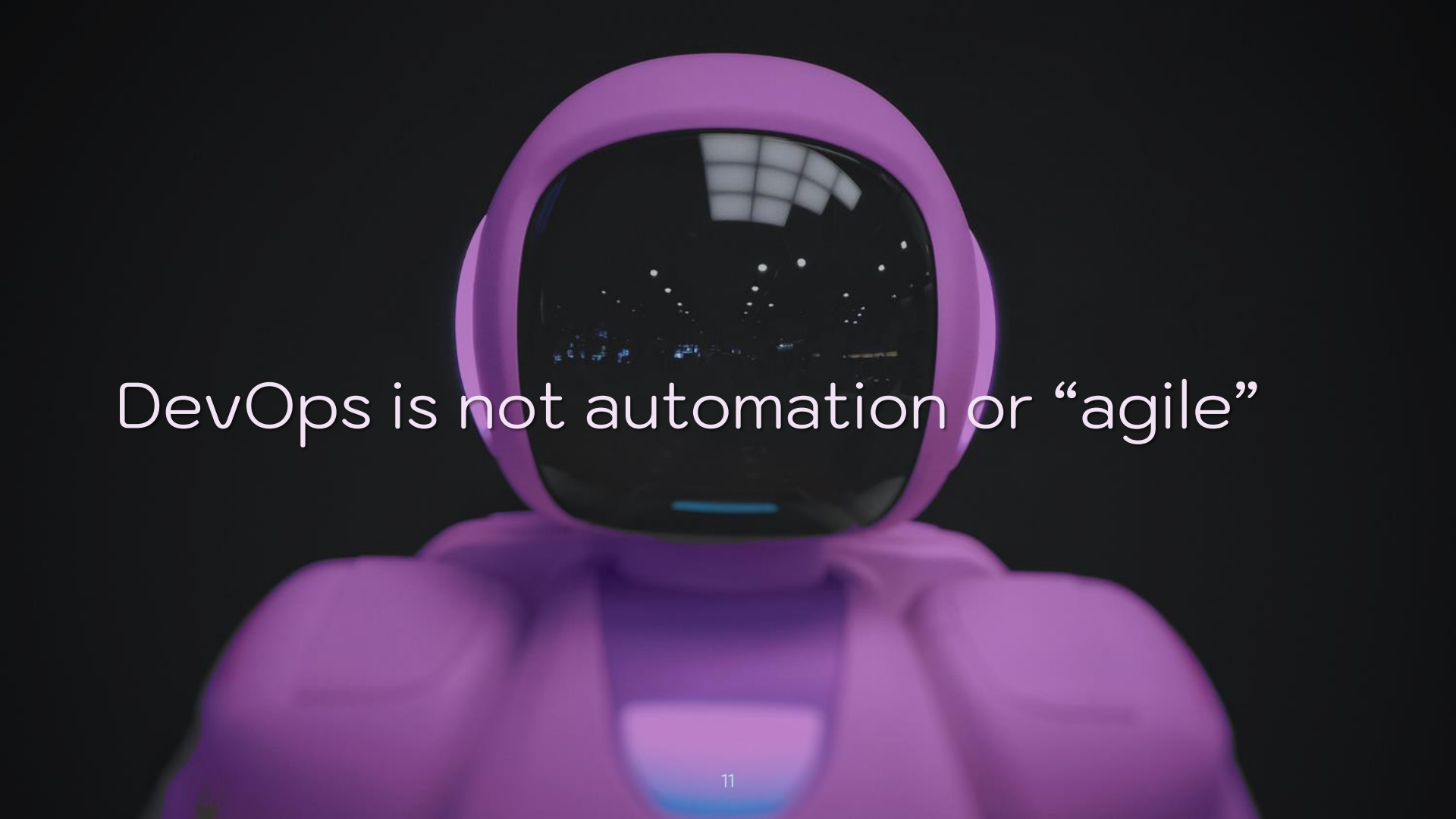
Infosec has a choice: marry DevOps or be rendered impotent & irrelevant


Infosec isn’t invincible. Denial &
“DevSecOps” won’t save your future.


How should infosec control chaos & make a marriage to DevOps last?

1. DevOps Dominion
2. The Metamorphosis
3. Time to D.I.E.
4. A Phoenix Rises
5. Marriage Vows


DevOps Dominion


DevOps is not automation or “agile”

DevOps is a mindset that unifies responsibility and accountability.


DevOps has “crossed the chasm” –
the business benefits are too striking

DevOps integrates once-disparate roles, encouraging “shifting left”


Infosec can join DevOps or watch as
DevOps carves its own secure path

Chaos & resilience is infosec's future

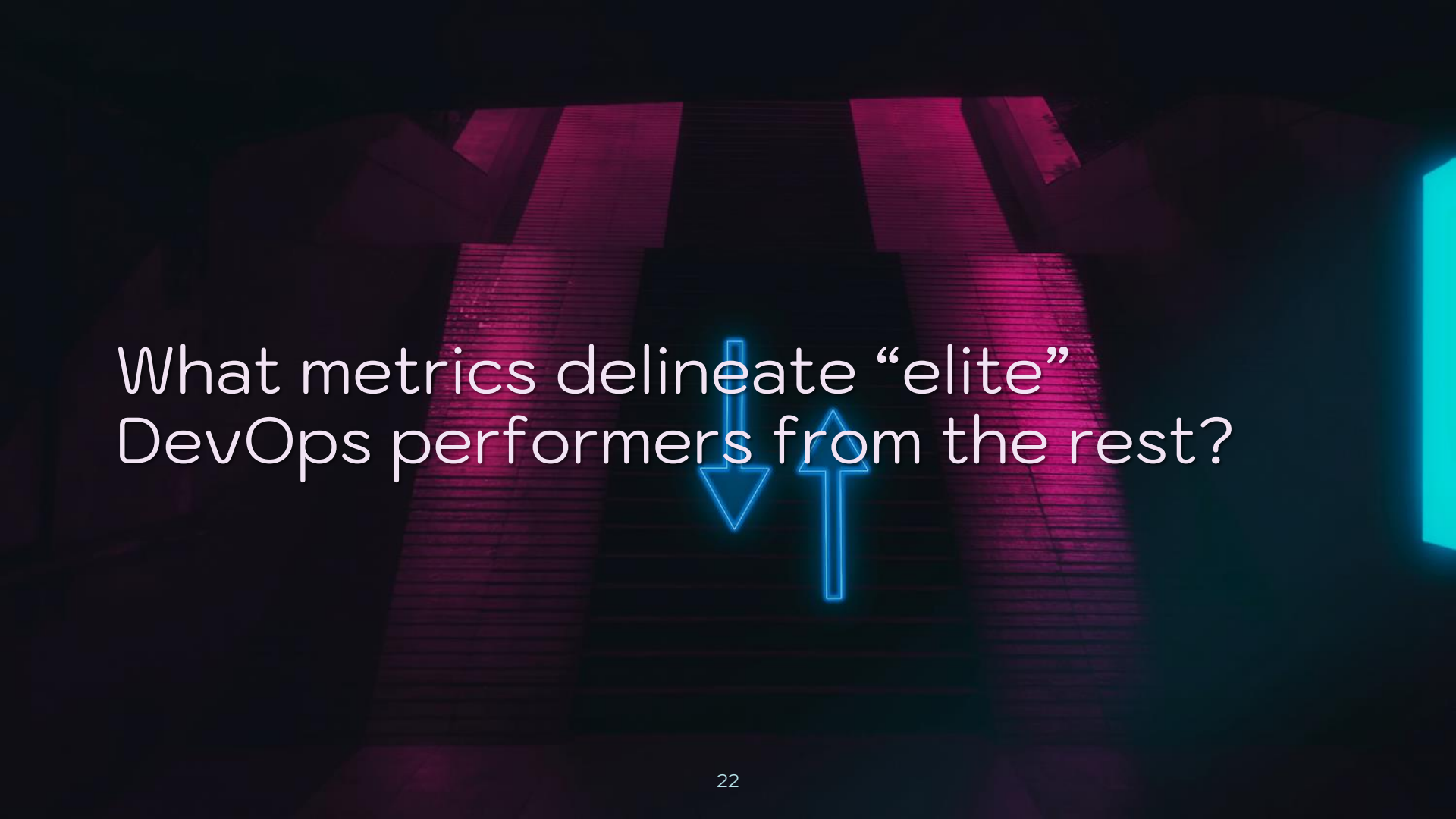
Therefore, infosec & DevOps priorities actually align...


What are DevOps's priorities?

Optimization of software delivery performance so tech delivers value


Stability & speed don’t conflict –
resilience & innovation are bffs

CI/CD: implement changes in prod rapidly, sustainably, & safely


What metrics delineate “elite”
DevOps performers from the rest?

Lead time for changes: How long does it take for committed code to successfully run in production?

Release frequency: How often is code deployed to production or released to end users?

Time to Recovery (TTR):

How long does it take to restore service?

Change failure rate: What percentage of changes to production degrade service & require remediation?

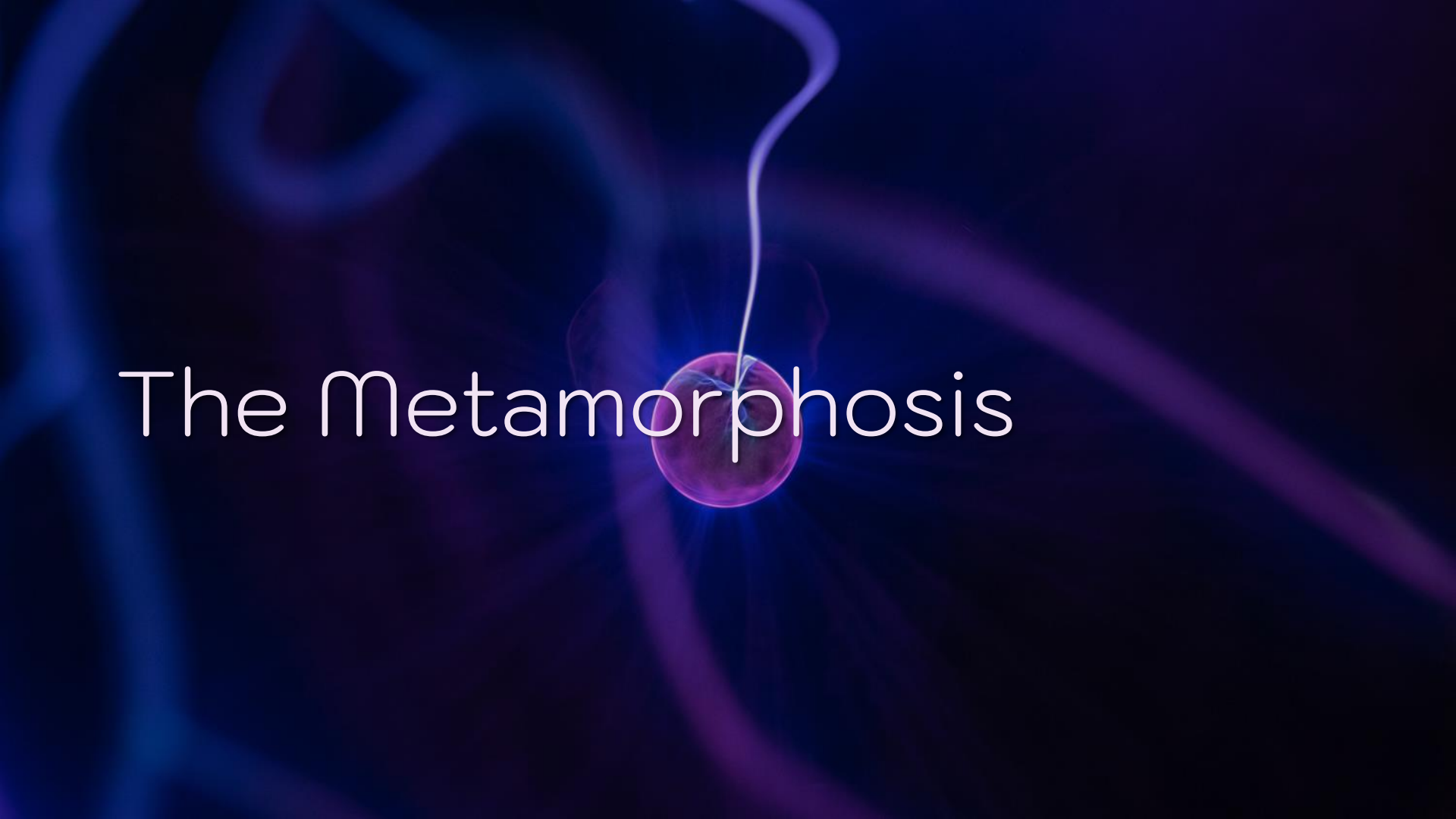
	Elite	High	Medium	Low
Lead time for changes	< One day	1 day - 1 week	1 week – 1 month	1 month – 6 months
Release frequency	On demand (>1 daily)	1 per day – 1 per month	1 per week – 1 per month	1 per month – 1 per 6 months
Time to recovery	< 1 hour	< 1 day	< 1 day	1 week – 1 month
Change failure rate	0% – 15%	0% – 15%	0% – 15%	46% – 60%


The evidence: no tradeoff between better infosec & DevOps leetness

Elites conduct security reviews & implement changes in mere days

“DevOps doesn’t care about security” is a lazy straw man. Stop it.


Security drives stronger DevOps results. Now infosec must evolve.


The Metamorphosis

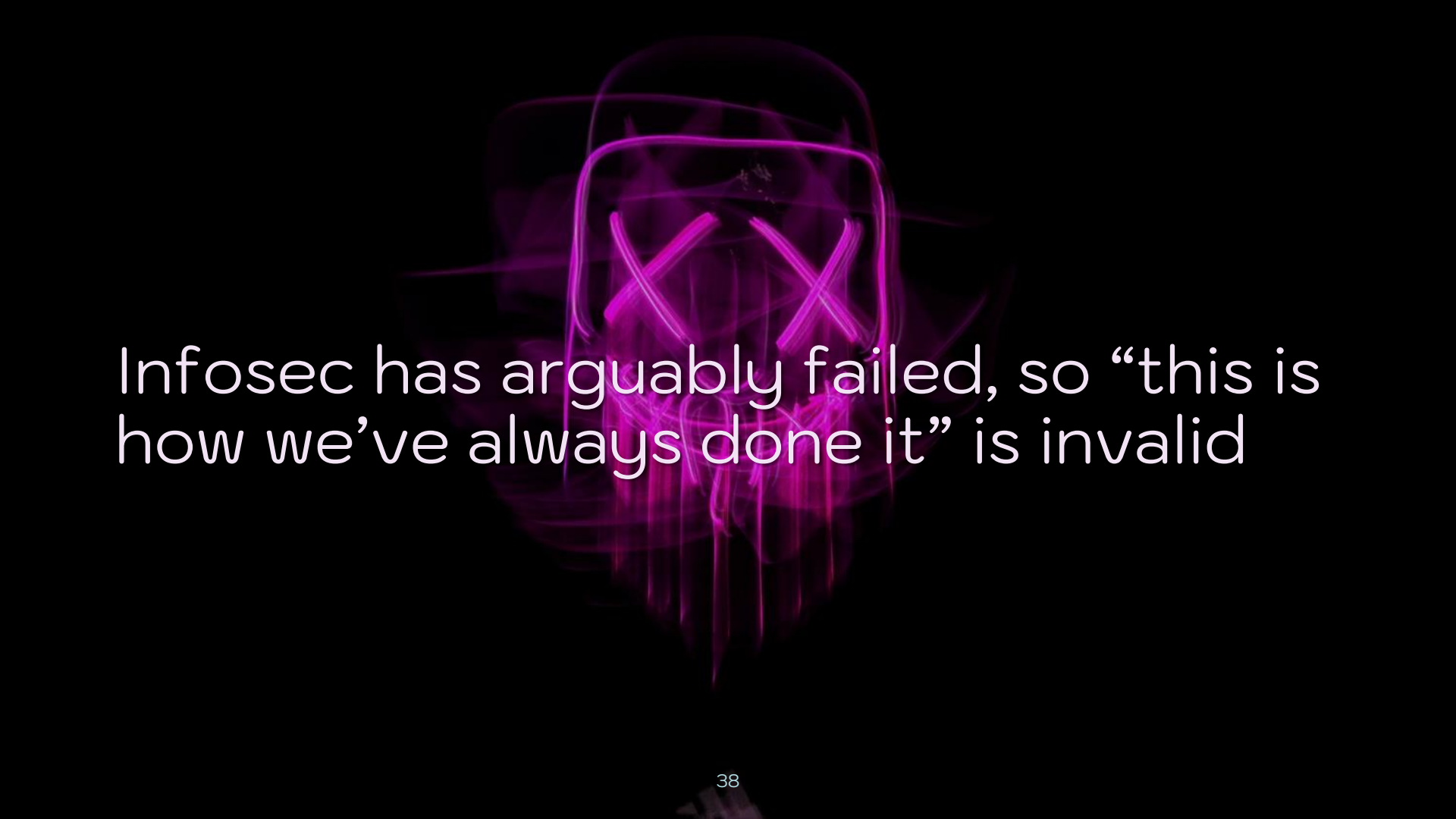
Partitioning of responsibility & accountability engenders conflict

The real “DevSecOps”: DevOps will be held accountable for security fixes


What goals should infosec pursue in this evolution?

And... why should infosec goals diverge from DevOps goals?

Infosec should support innovation in the face of change – not add friction



Infosec has arguably failed, so “this is how we’ve always done it” is invalid

Cloud & microservices created the “Infosec Copernican Revolution”

But the data doesn’t lie: cloud & PaaS contribute to elite performance

The Security of Chaos

“Things will fail” naturally extends into “things will be pwned”

Security failure is when security controls don’t operate as intended


What are the principles of chaotic security engineering?

1. Expect that security controls will fail & prepare accordingly

2. Don’t try to avoid incidents – hone your ability to respond to them

What are the benefits of the chaos / resilience approach?

Benefits: lowers remediation costs & stress levels during real incidents

Benefits: minimizes end-user disruption & improves confidence

Benefits: creates feedback loops to foster understanding of systemic risk

The ability to automate “toil” away should also appeal to infosec


Toil: manual, repetitive, tactical work
that doesn't provide enduring value

Manual patching, provisioning 2FA / ACLs, firewall rule management, etc.

What other ways can infosec become more strategic?


Time to D.I.E.

C.I.A. triad: commonly used as a model to balance infosec priorities

Confidentiality: withhold info from people unauthorized to view it

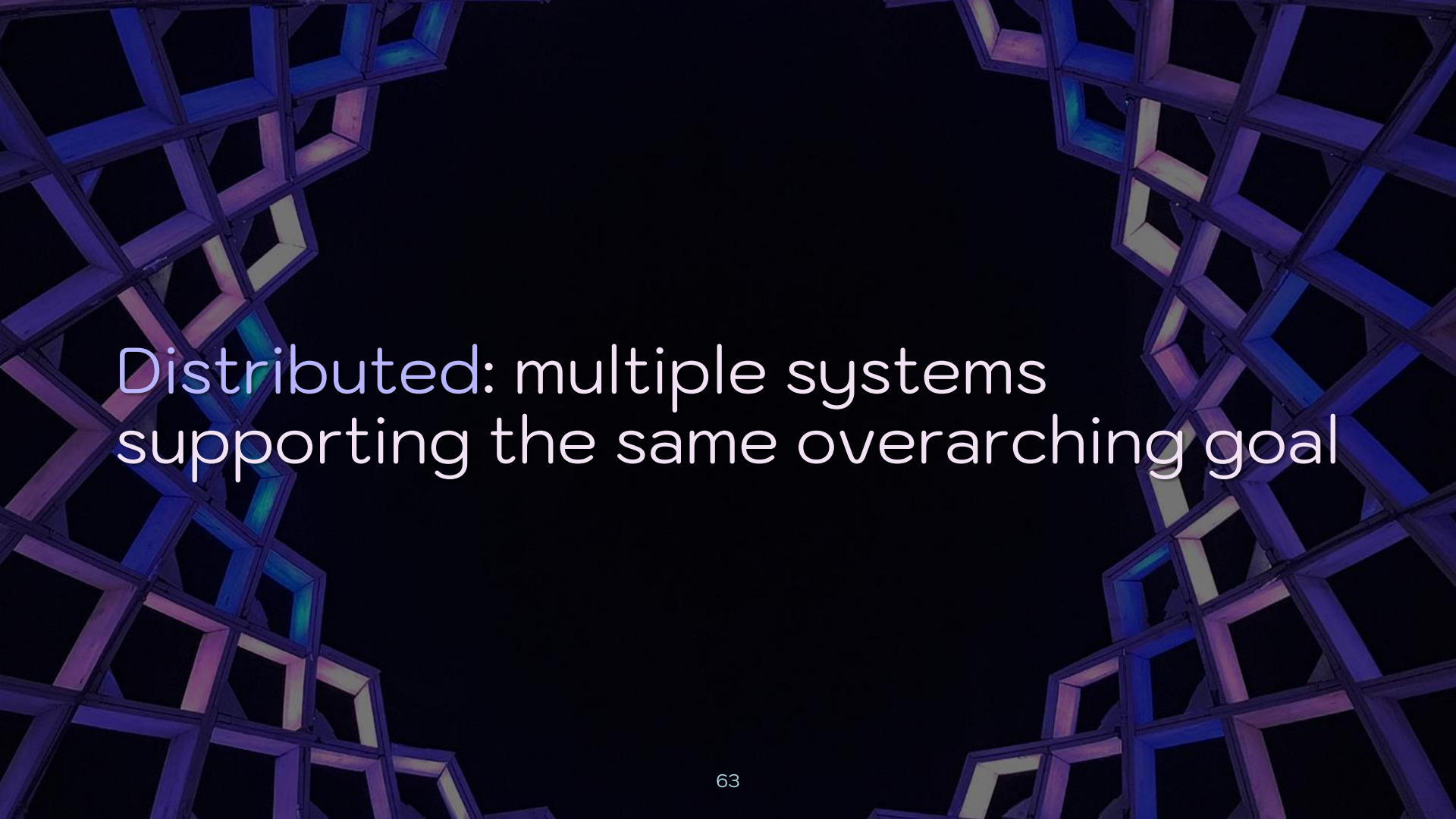
Integrity: data is a trustworthy representation of the original info

Availability: organization's services are available to end users


But these are security *values*, not qualities that create security

We need a model promoting qualities that make systems more secure

Instead, use the D.I.E. model:
Distributed, Immutable, Ephemeral


Distributed: multiple systems
supporting the same overarching goal

Distributed infrastructure reduces risk of DoS attacks by design


Immutable: infrastructure that
doesn't change after it's deployed

Servers are now disposable “cattle” rather than cherished “pets”


Immutable infra is more secure by
design – ban shell access entirely

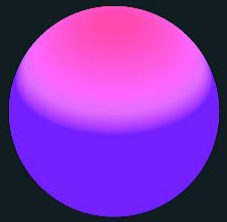
Lack of control is scary, but unlimited lives are better than nightmare mode


Ephemeral: infrastructure with a very short lifespan (dies after a task)

Ephemerality creates uncertainty for attackers (persistence = nightmare)

Installing a rootkit on a resource that dies in minutes is a waste of effort

Optimizing for D.I.E. reduces risk by design & supports resilience

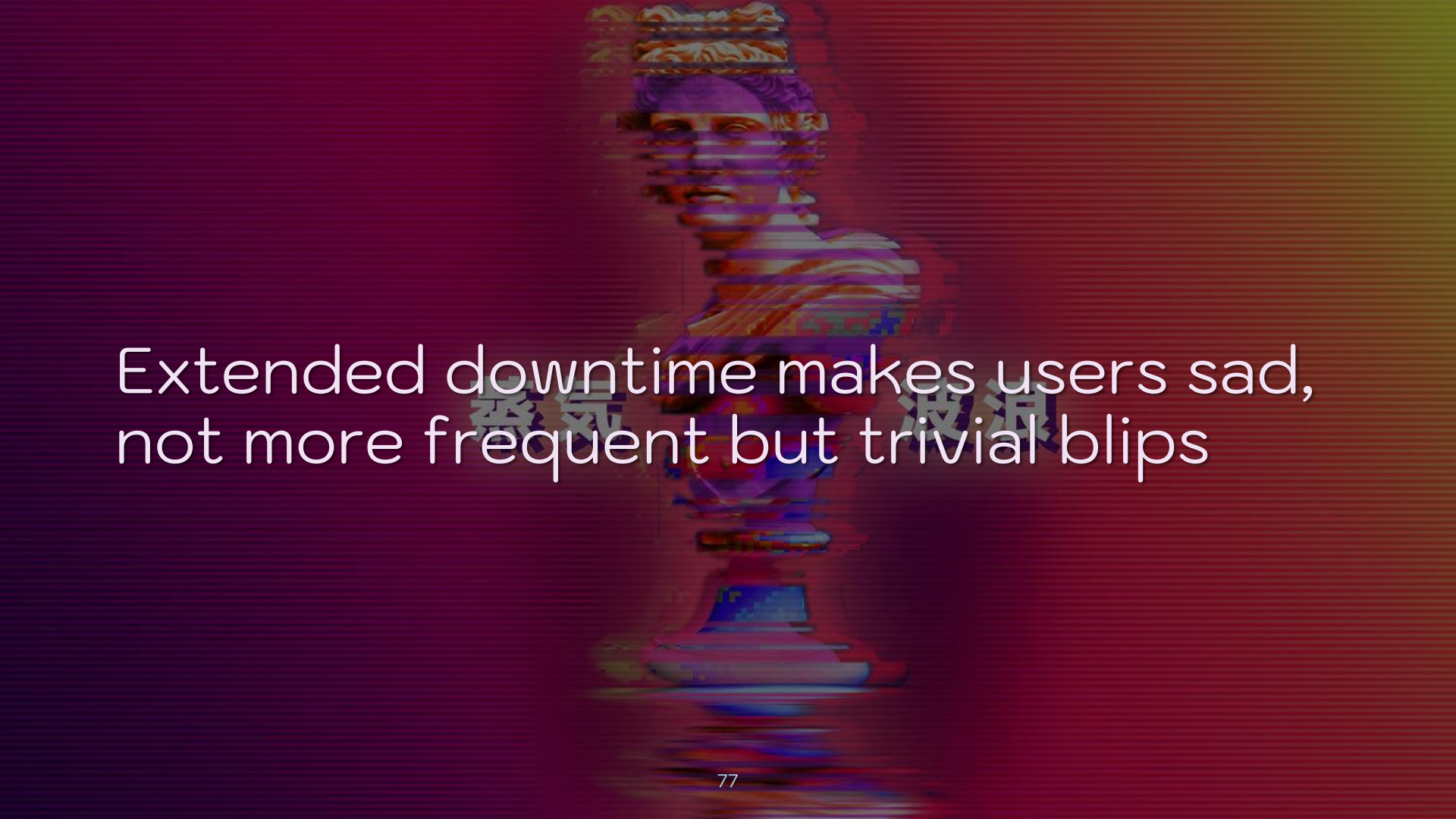


A Phoenix Rises

What metrics support resilient security engineering?


TTR is equally as important for infosec as it is for DevOps

Time Between Failure (TBF) will lead your infosec program astray



Extended downtime makes users sad,
not more frequent but trivial blips

Prioritizing failure inhibits innovation

Instead, harness failure as a tool to help you prepare for the inevitable

TTR > TTD – who cares if you detect quickly if you don’t fix it?


Game days: like planned firedrills

Prioritize game days based on potential business impacts


Decision trees: start at target asset, work back to easiest attacker paths

Determine the attacker's least-cost path (hint: it doesn't involve 0day)


Architecting chaos

Begin with “dumb” testing before moving to “fancy” testing

Controlling Chaos: Availability

Turning security events into availability events appeals to DevOps


The existing repertoire of chaos eng tools primarily covers availability

Chaos Monkey, Azure Fault Analysis Service, Chaos-Lambda...

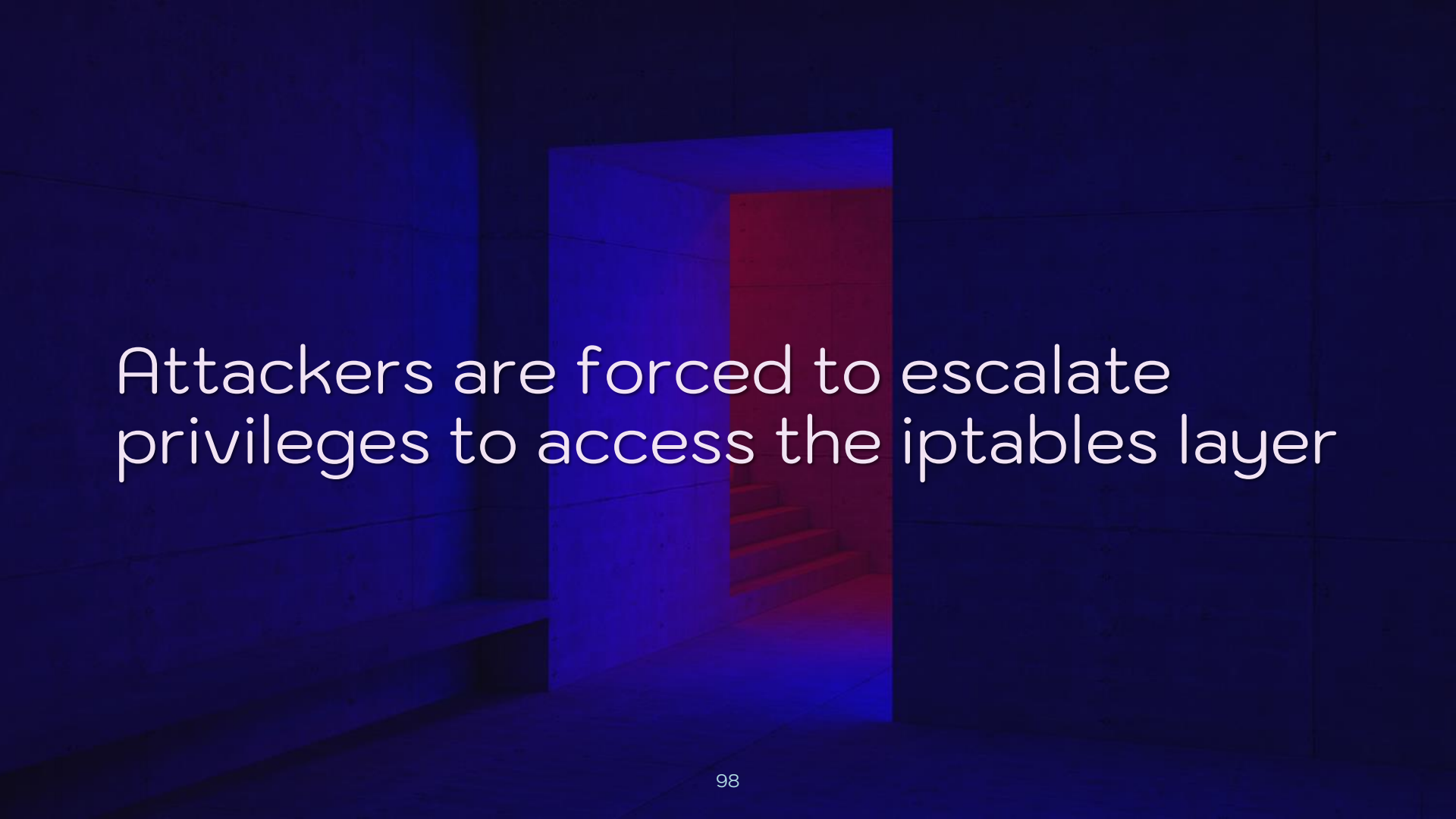
Kube-monkey, PowerfulSeal, Pod-reaper, Pumba, Blockade...

Infosec teams can use these tools but make attackers the source of failure

Controlling Chaos: Confidentiality

Microservices use multiple layers of auth that preserve confidentiality

A service mesh is like an on-demand VPN at the application level


Attackers are forced to escalate
privileges to access the iptables layer

Test: inject failure into your service mesh to test authentication controls


Controlling Chaos: Integrity

Test: swap out certs in your ZTNs – all transactions should fail

Test: modify encrypted data & see if your FIM alerts on it

Test: retrograde libraries, containers, other resources in CI/CD pipelines


D.I.E.ing is an art, like everything else


Controlling Chaos: Distributed

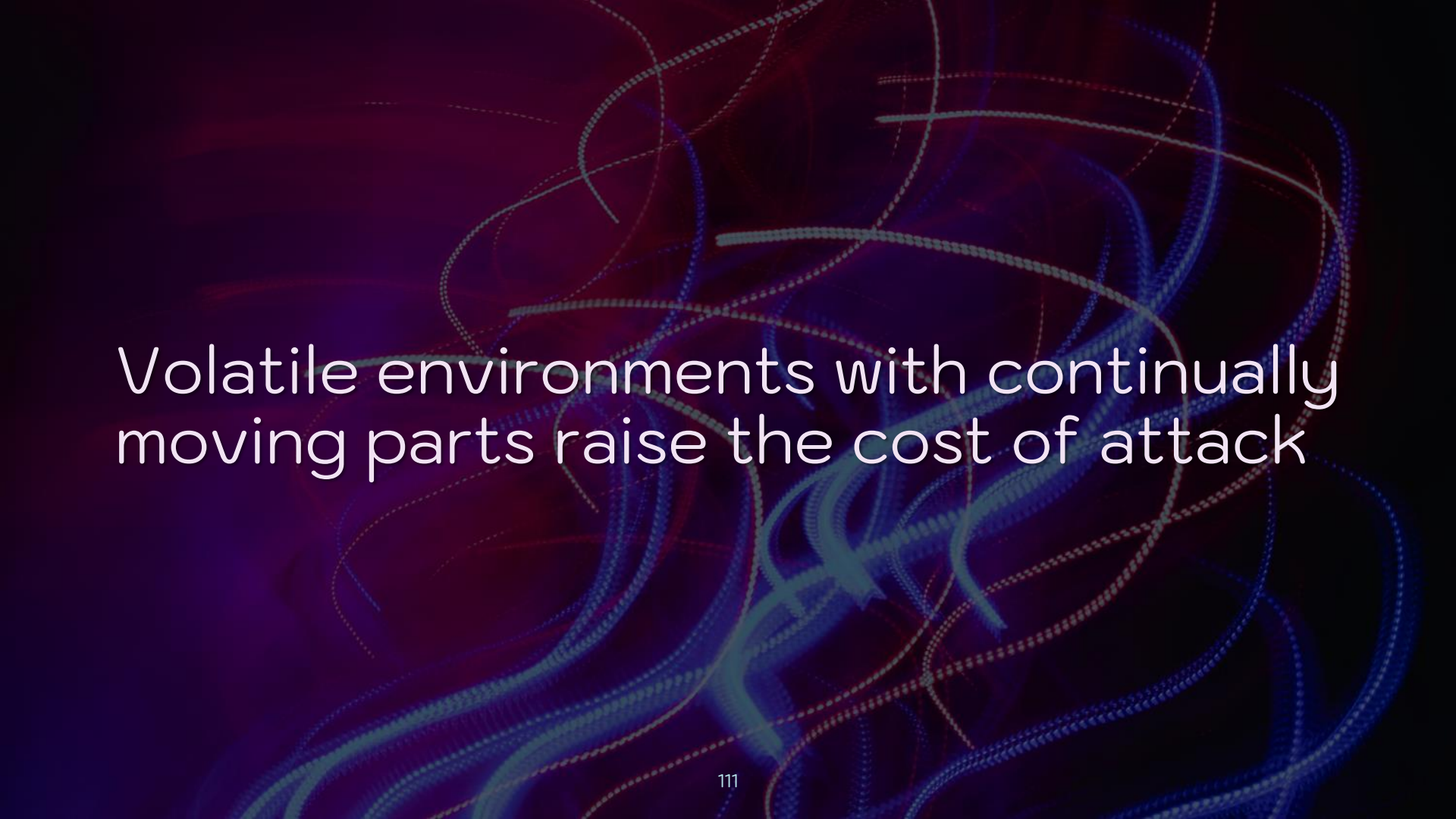
Distributed mostly overlaps with availability in modern infra contexts


Multi-region services present a fun opportunity to mess with attackers

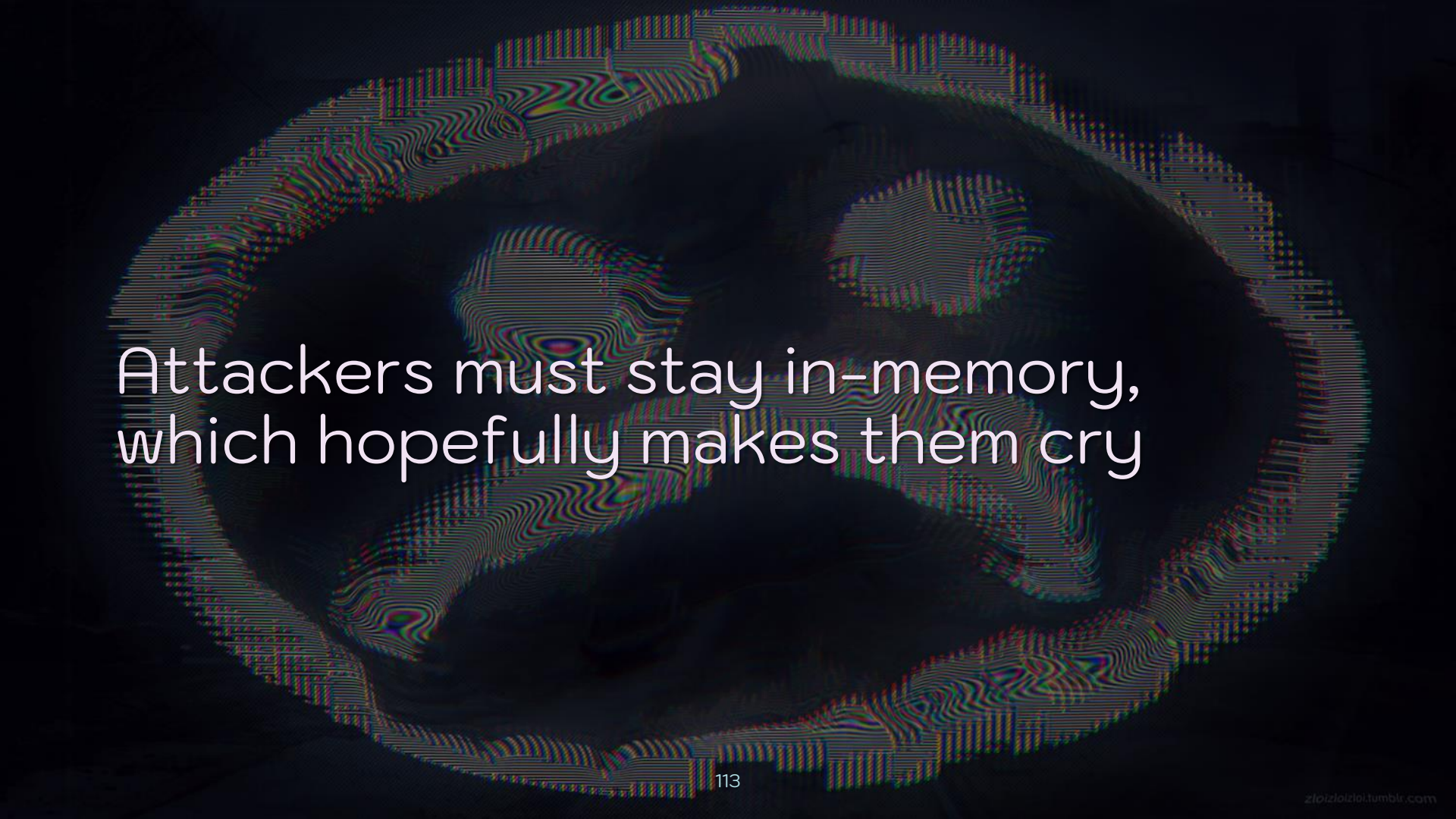
Shuffle IP blocks regularly to change attackers' lateral movement game


Controlling Chaos: Immutable

Immutable infra is like a phoenix – it disappears & comes back a lot


Volatile environments with continually moving parts raise the cost of attack

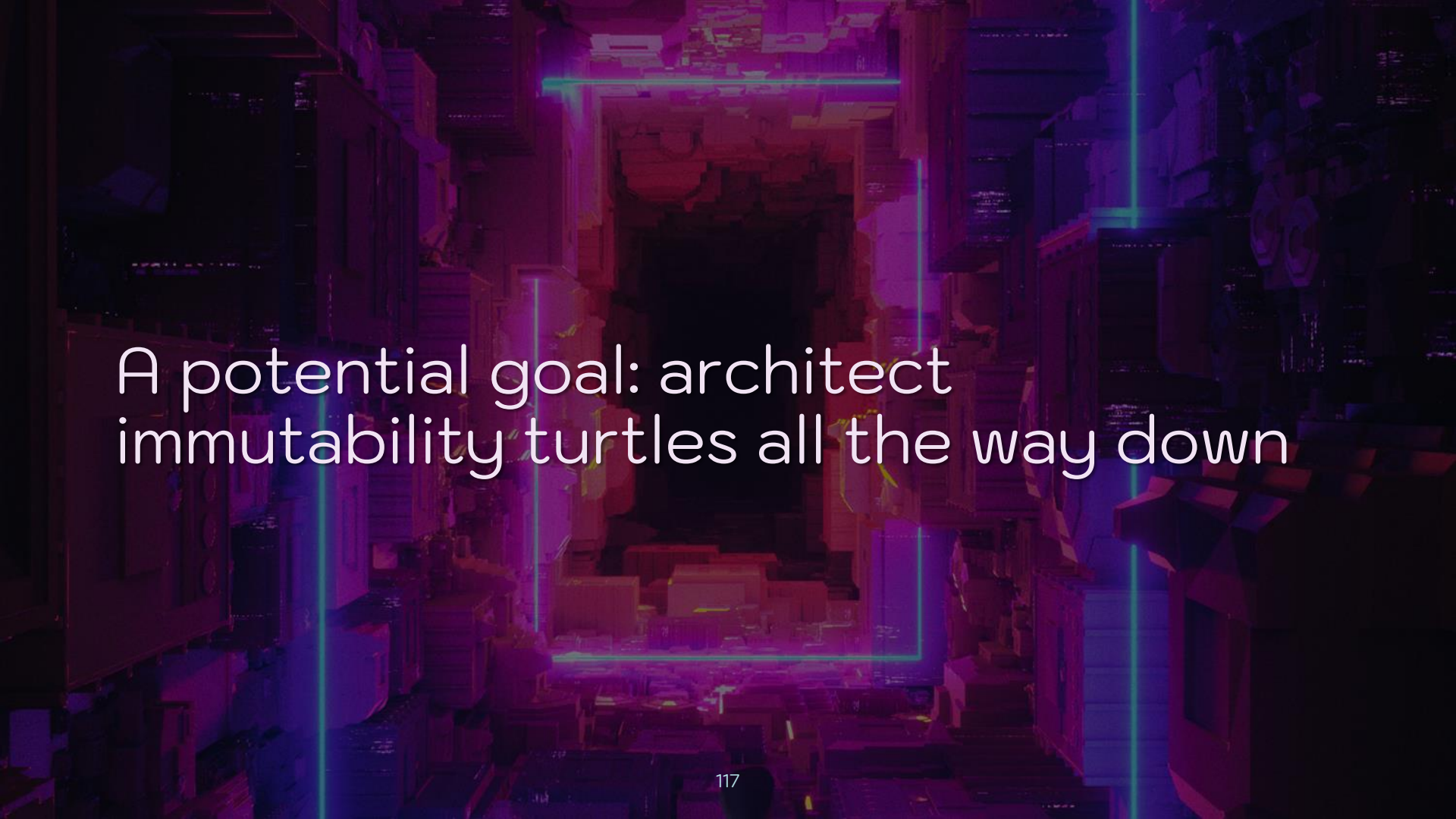
Create rules like, “If there’s ever a write to disk, crash the node”



Metasploit Meterpreter + webshell: Touch passwords.txt & kaboom


Infosec teams can build Docker images with a “bamboozle layer”

Mark garbage files as “unreadable” to craft enticing bait for attackers


A potential goal: architect
immutability turtles all the way down

Test: inject attempts at writing to disk to ensure detection & reversion

Treat changes to disk by adversaries similarly to failing disks: mercy kill


Controlling Chaos: Ephemeral

Most infosec bugs are stated-related – get rid of state, get rid of bugs

Reverse uptime: longer host uptime adds greater security risk

Test: change API tokens & test if services still accept old tokens

Test: inject hashes of old pieces of data to ensure no data persistence

Use “arcade tokens” instead of using direct references to data

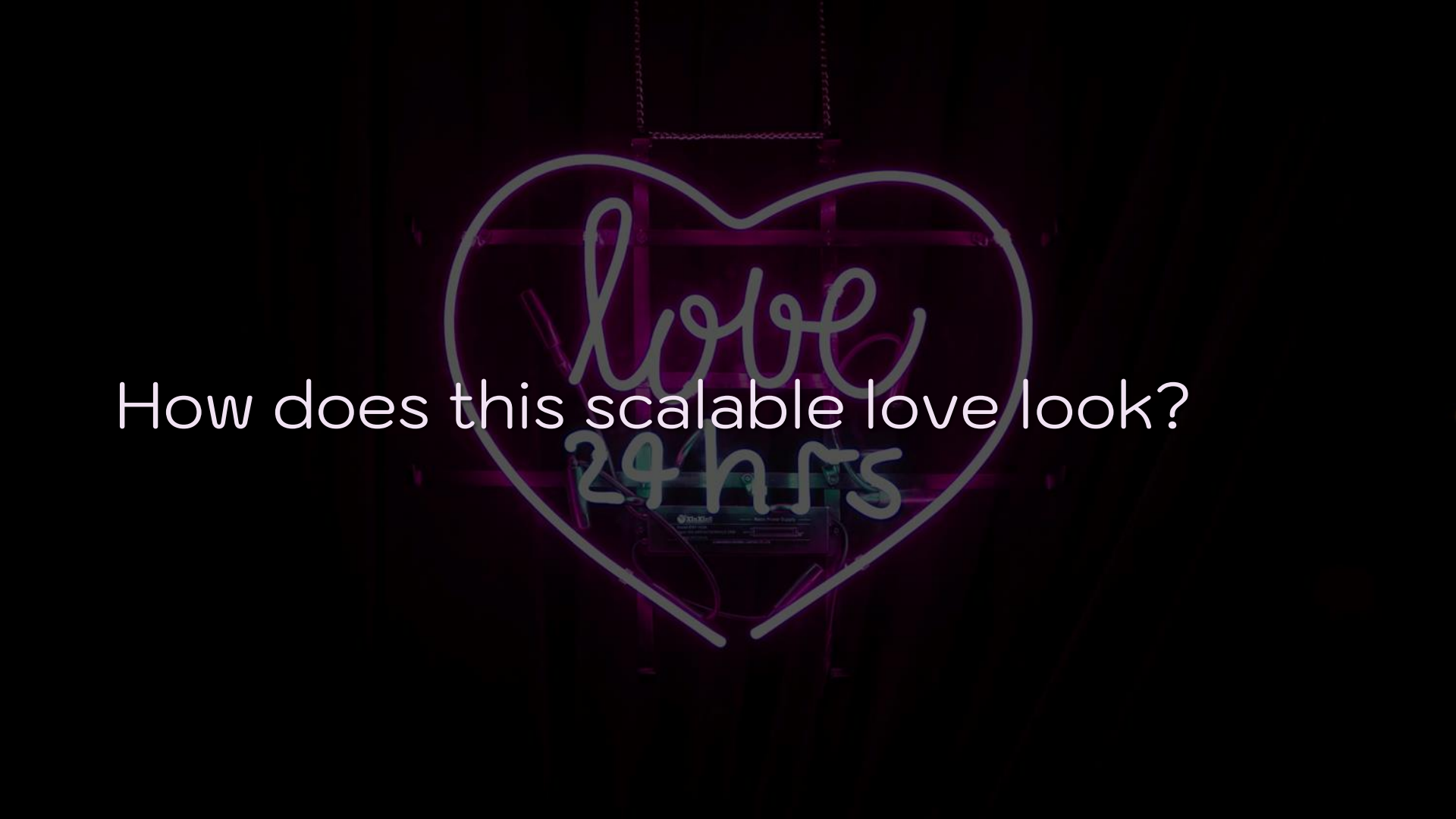
Leverage lessons from toll fraud – cloud billing becomes security signal

Test: exfil TBs or run a cryptominer to inform billing spike detection

So, how should infosec work with DevOps to implement all of this?

Marriage Vows

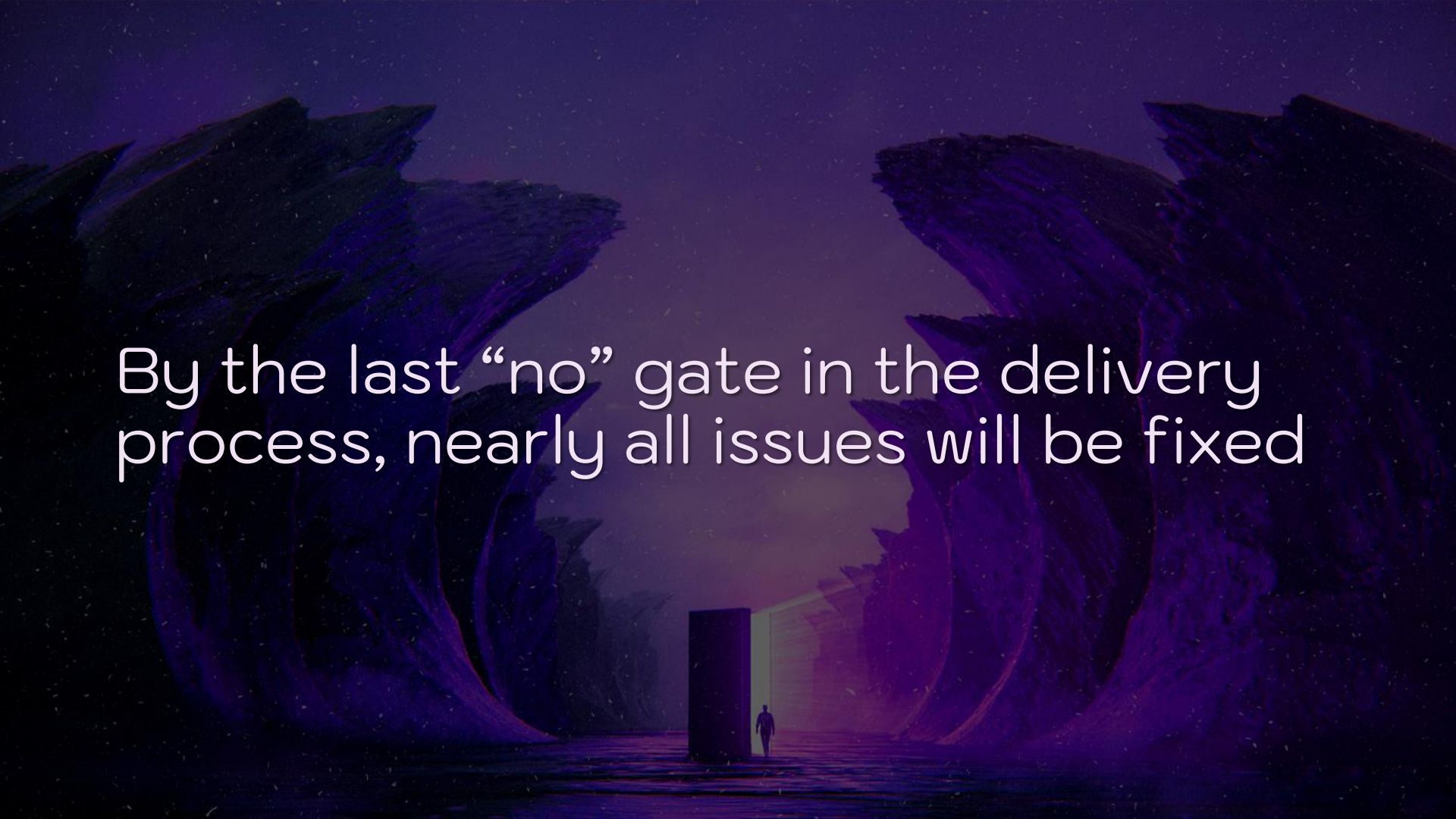
Infosec + DevOps = scalable love



How does this scalable love look?

Sit in on early design decisions & demos – but say “No, and…” vs. “No.”

Provide input on tests so every testing suite has infosec’s stamp on it

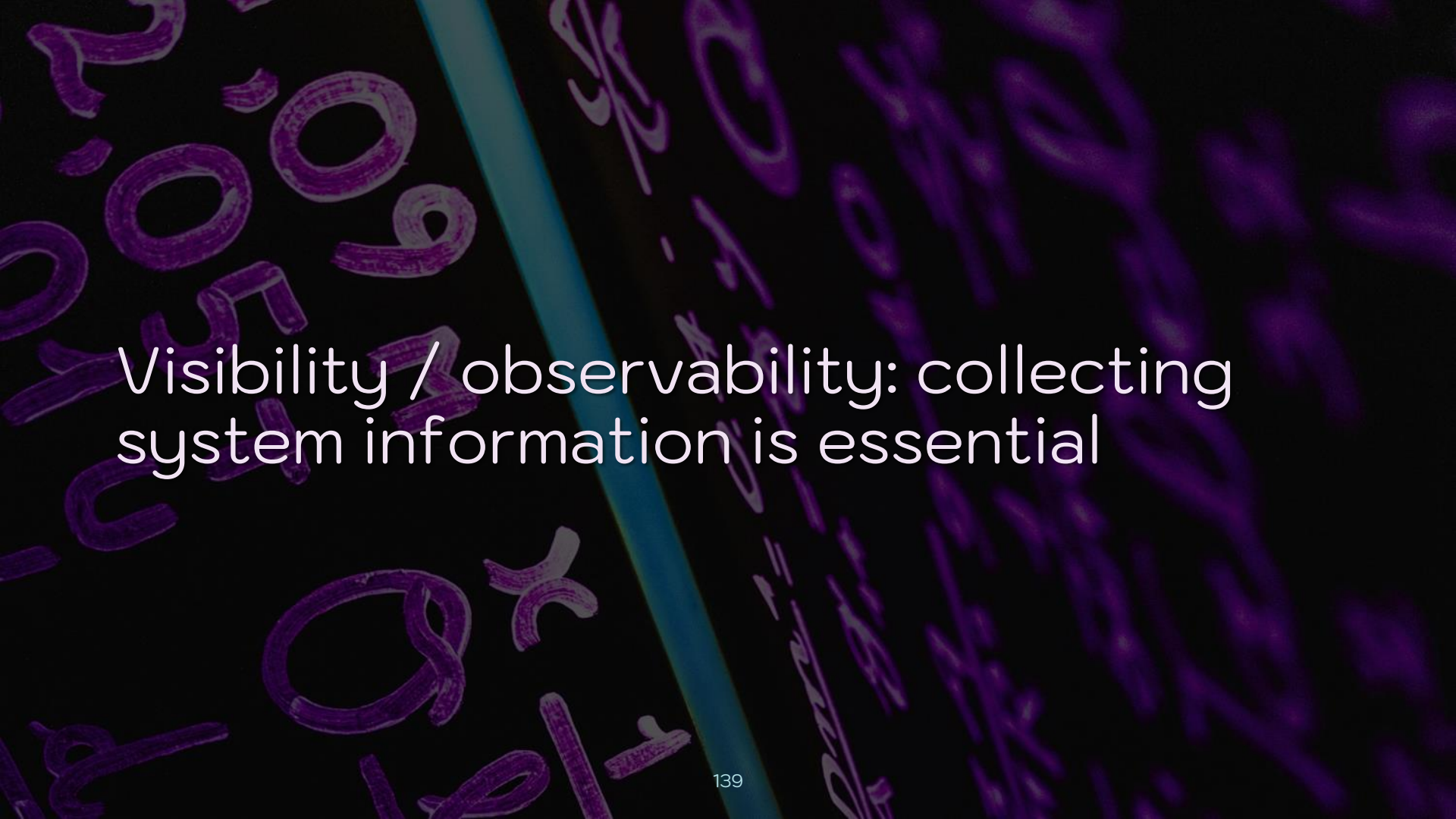

By the last “no” gate in the delivery process, nearly all issues will be fixed

Infosec should focus on outcomes that are aligned with business goals

TTR should become the preliminary anchor of your security metrics

Security- & performance-related gamedays can’t be separate species

Cultivate buy-in together for resilience & chaos engineering



Visibility / observability: collecting system information is essential

Your DevOps colleagues are likely already collecting the data you need

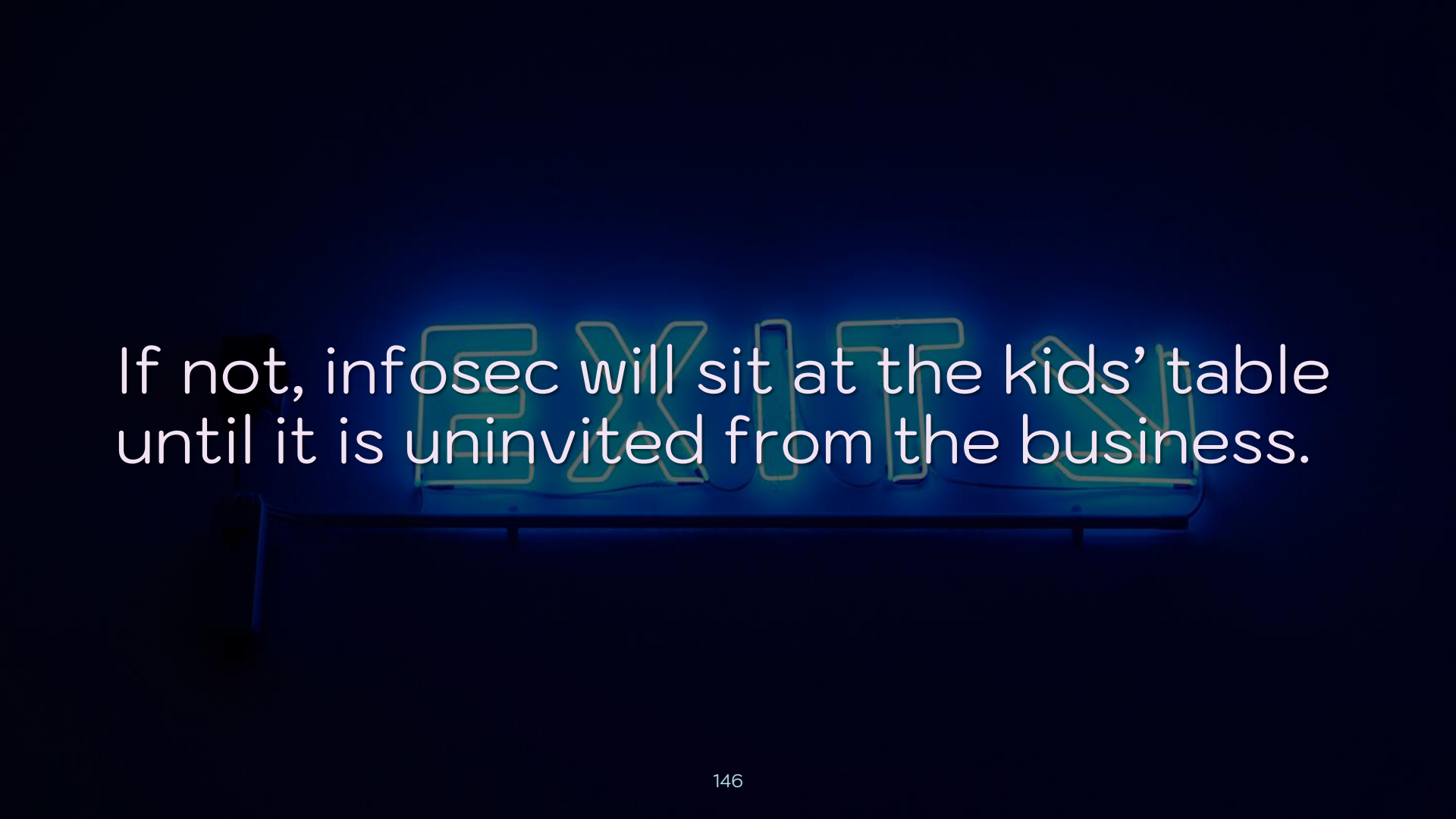
Changing culture: change what people do, not what they think

Conclusion

Security cannot force itself into DevOps. It must marry it.


Chaos/resilience are natural homes
for infosec & represent its future.


Infosec must now evolve to unify responsibility & accountability.



If not, infosec will sit at the kids' table until it is uninvited from the business.


Giving up control isn’t a harbinger of doom. Resilience is a beacon of hope.

“You must have chaos within you to give birth to a dancing star.”

— Friedrich Nietzsche

@swagitda_

@nicolefv

/in/kellyshortridge

/in/nicolefv

kelly@greywire.net

nicolefv@google.com