

# WATERING THE ROOTS OF RESILIENCE:

## LEARNING FROM FAILURE WITH DECISION TREES

Kelly Shortridge @swagitda\_ | shortridge@hachyderm.io

SREcon Americas 2023

A photograph of a desert landscape. In the foreground, several dead, black, gnarled trees stand on a flat, light-colored sandy plain. In the background, large, smooth, reddish-brown sand dunes rise against a clear, bright blue sky. The scene is brightly lit, suggesting a sunny day.

Systems resilience depends on the ability to adapt and evolve to changing conditions.



Our software systems are complex,  
sociotechnical, and difficult to reason about.

A landscape photograph of a valley at sunset. The foreground features a circular stone fire pit in the center, surrounded by a dirt path. The valley is filled with green grass and scattered rocks. In the background, a large, craggy rock formation stands on the right, and the sun is setting between two hills, casting a warm glow over the scene. The sky is filled with soft, wispy clouds.

Humans are the mechanism for adaptation in our complex software systems.



How can SREs align their mental models of the system with reality to sustain resilience?

I. Adaptation in Complex Systems

II. Resilience Stress Testing with Chaos Experiments

III. Refining Mental Models with Decision Trees

A lush garden scene featuring a waterfall cascading over large, dark grey rocks into a pond. The foreground is filled with various green plants, including tall grasses and irises. The background is dominated by dense evergreen trees, creating a rich, green environment. The lighting is soft, suggesting late afternoon or early morning.

# I. Adaptation in Complex Systems

Complex systems present a large variety of possible states; prediction is impossible.



A white cat with blue eyes is crouching in a grassy area, looking towards the left. The background is blurred green grass and foliage. The text is overlaid on the left side of the image.

Getting from point A to point B in complex systems is more as a cat zoomies vs. crow flies.

The reality is failure is inevitable; it's a natural part of complex systems as they operate


A large octopus with a mottled brown and tan pattern is resting on a rocky seabed underwater. The octopus's head is at the top center, and its thick, wrinkled arms extend downwards and outwards. The background is a clear, deep blue ocean. The text "Complex systems are adaptive: they evolve in response to changes in their environment." is overlaid in white, sans-serif font across the middle of the image.

Complex systems are adaptive: they evolve in response to changes in their environment.

**Adaptive capacity:** how poised a system is to change how it works based on context

Resilience is “the ability to prepare and plan for, absorb, recover from, and more successfully adapt to adverse events.”


What is the resilience potion recipe? There are five ingredients to sustain resilience...

A glass of milk with a straw, splashing against a dark background. The milk is captured in mid-air, creating a dynamic splash around the glass. The lighting is dramatic, highlighting the white milk against the dark background.

Define the system's  
critical functions

Define the system's  
safe boundaries



A photograph of a brick wall with a shadow cast by a brick on the left side. The bricks are arranged in a grid pattern, and the shadow is cast by a brick in the top-left corner. The text "Observe system interactions across spacetime" is overlaid on the image in a white serif font.

Observe system  
interactions across  
spacetime



Feedback loops and a  
learning culture

A close-up photograph of a dark green ceramic mug filled with a dark liquid, likely hot chocolate. The top is covered with several white marshmallows, some of which are dusted with a brown powder, possibly cinnamon. A single cinnamon stick is placed vertically in the mug. The background is out of focus, showing several warm, yellow bokeh lights. The overall mood is cozy and festive.

Flexibility and  
willingness to change

How does this look in our computer systems?  
What is “adaptation” in them?

Our software systems have machines and humans continually influencing each other.

An elephant is shown in profile, facing right, in a natural savanna environment. It is actively spraying a large amount of reddish-brown dust from its trunk, creating a thick, billowing cloud of dust that rises and drifts to the right. The elephant's trunk is curled upwards, and its tusks are visible. The background consists of various green and brown trees and bushes under a clear sky. The ground is dry and dusty.

When machine processes fail in ways that are noticeable, humans jump in.

Software has limited ability to adapt on its own.  
Humans are the primary adaptive capability.

A photograph of four turtles resting on a log in a pond. The turtles are arranged along the log, with their reflections visible in the water below. The background is a lush green pond with some reeds or grasses visible.

Consider Log4Shell: real-world harm was low due to the socio part of the system.

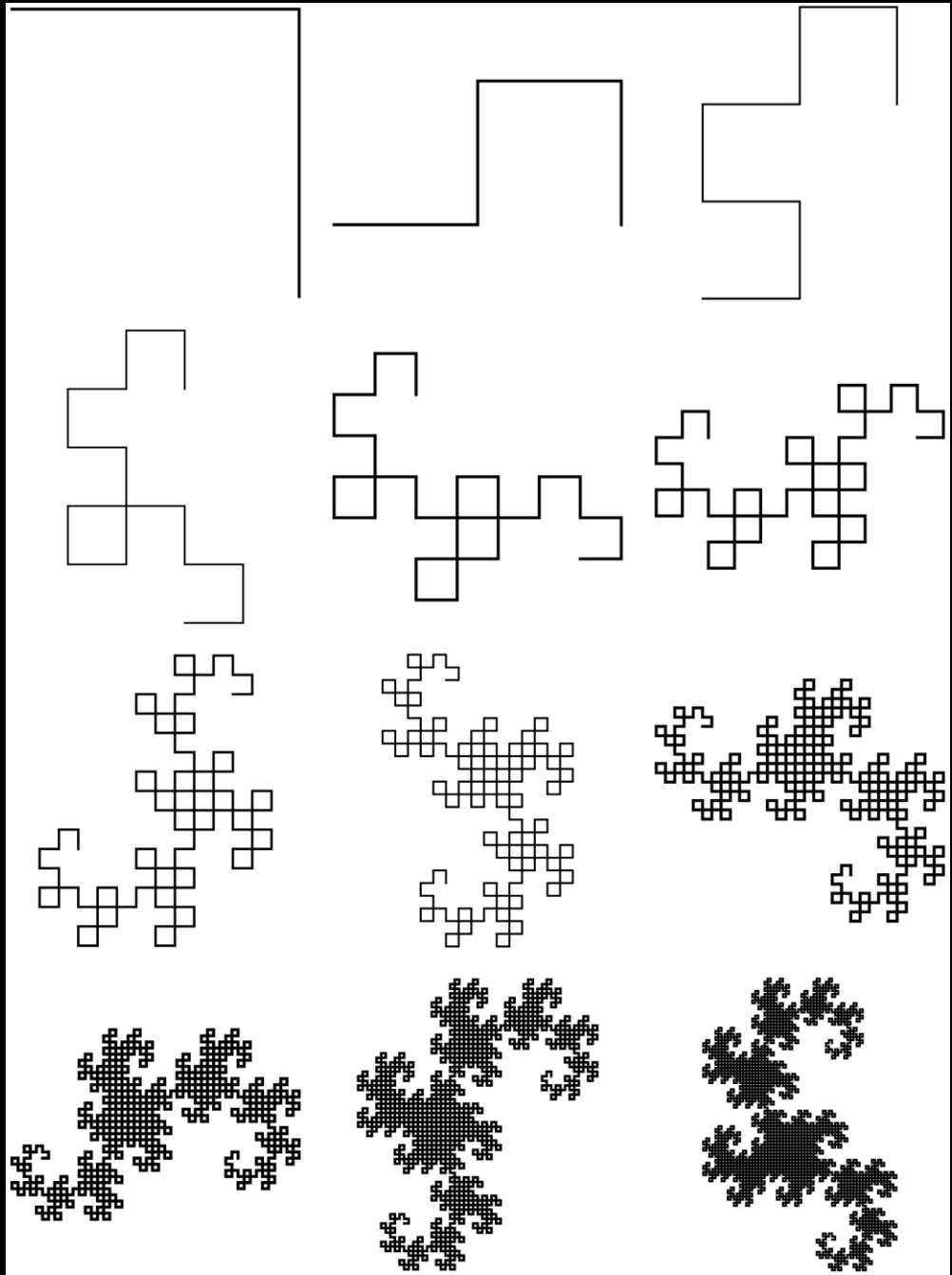


How we learn influences how we adapt to stressors, surprises, and adverse conditions.

A close-up photograph of a sloth's face, showing its eyes and nose, with the text overlaid.

The software we design, build, and operate reflects our mental models of reality.

Naturally, our mental representations of reality are incomplete and inconsistent.



**Surprise** is “the revelation that a given phenomenon of the environment was, until this moment, misinterpreted.”

A close-up photograph of two blue parrots with yellow beaks and yellow-ringed eyes, peering out from a circular hole in a tree trunk. The tree trunk is made of rough, textured bark. The background shows green foliage on the left and right sides.

We must “prepare to be surprised.”

A dramatic photograph of a volcanic eruption. A large plume of dark smoke and ash rises from a mountain peak. In the foreground, a massive flow of bright orange and red lava cascades down the slope. A single, bright purple lightning bolt strikes the center of the lava flow, creating a stark contrast with the fiery scene. The background is a dark, overcast sky.

## II. Resilience Stress Testing with Chaos Experiments

Our goal is to uncover “baffling interactions” in our systems that defy our expectations.





We can do so through chaos experiments:  
resilience stress tests for software systems.

Experiments can generate evidence of how much our mental models deviate from reality.



Chaos experiments help us more quickly learn about system behavior and its context.

We conduct system-level adverse scenarios rather than evaluating specific components.

A dramatic landscape featuring a stormy sky with dark, heavy clouds and a bright lightning bolt striking down. The scene is reflected in a calm body of water in the foreground. To the right, a dark, rocky hillside with sparse vegetation rises. The overall mood is somber and powerful.

Many weaknesses only emerge once the system is, in effect, a living thing...

We can *fix* things in production by learning from adverse conditions via experiments.



We're curious about assessing the nature of the system and its interconnections.

How are chaos experiments different than any other kind of test?



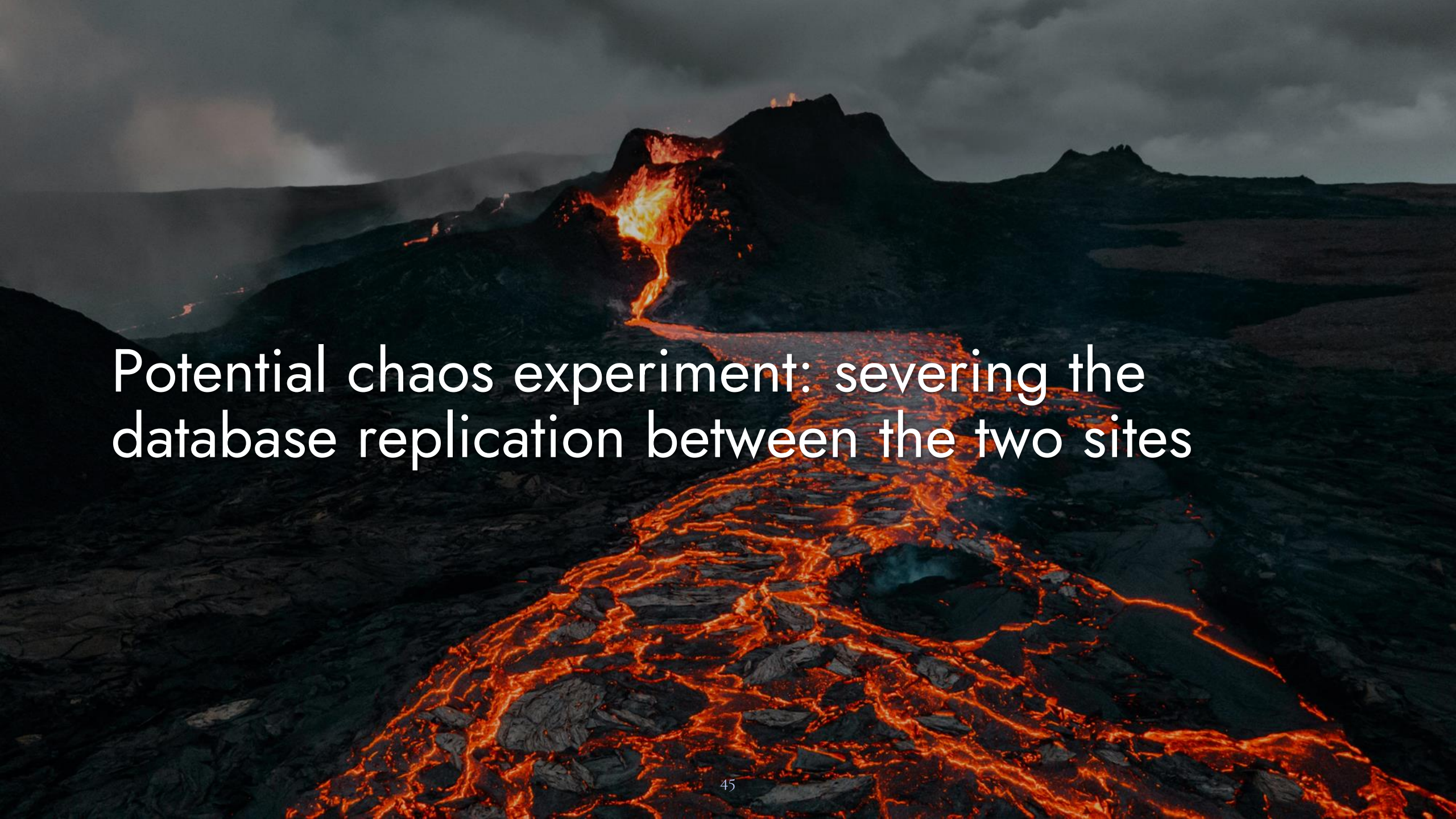
Chaos experiments	Typical tests
Support resilience	Support robustness
Sociotechnical (includes humans as part of the system)	Technical (excludes humans as part of the system)
System-focused	Component-focused
Capable of being run in production	Must run in dev or staging
Random or pseudo-random	Scripted
Adverse scenarios	Boolean requirements
Observe and learn from failures	Detect and prevent failures
N-order effects	Direct effects

A wide-angle shot of a modern industrial factory floor. The scene is dominated by a central orange robotic arm mounted on a white base, positioned within a large white metal frame. The floor is dark and reflective, with yellow safety lines. In the foreground, there are several stacks of cardboard boxes on pallets, including brands like 'Ganten' and 'Red Bull'. The background shows a vast, high-ceilinged space with multiple levels, walkways, and other industrial equipment. The lighting is bright and even, highlighting the metallic surfaces and the organized layout of the facility.

# Interlude: a Case Study


Example: physical parts database for its manufacturing sites on both coasts in the USA

How do their systems behave when replication is severed or when it gets too far behind?



Potential chaos experiment: severing the  
database replication between the two sites

Hypothesis proven incorrect: replication didn't work; requests still served from the West Coast



Design change: halting the west coast data-center if it isn't caught up to the primary

Re-run the experiment: replication was faster than expected (yay!) but no alerts fired (oof!)



Disruptions can include security issues, too, since they can become reliability problems.

A photograph of a pond with numerous lily pads floating on the surface. The lily pads are in various stages of growth and color, ranging from green to brown and red. Several bright orange koi fish are swimming in the dark water between the lily pads. The scene is captured from a high angle, looking down into the pond.

Attackers love to take advantage of interactions between components to compromise a system.

A large flock of albatrosses is shown in flight over a turbulent, stormy ocean. The waves are dark blue and green with white foam, and the sky is overcast and grey. The birds are scattered throughout the scene, some in the foreground and others in the distance, creating a sense of a vast, chaotic environment.

# Introduction to Security Chaos Engineering

**Security Chaos Engineering (SCE):**  
a socio-technical transformation that enables the organizational ability to gracefully respond to failure and adapt to evolving conditions.



SCE aligns mental models with reality and improves our systems' resilience to attack.

How do we create a security chaos experiment in practice?

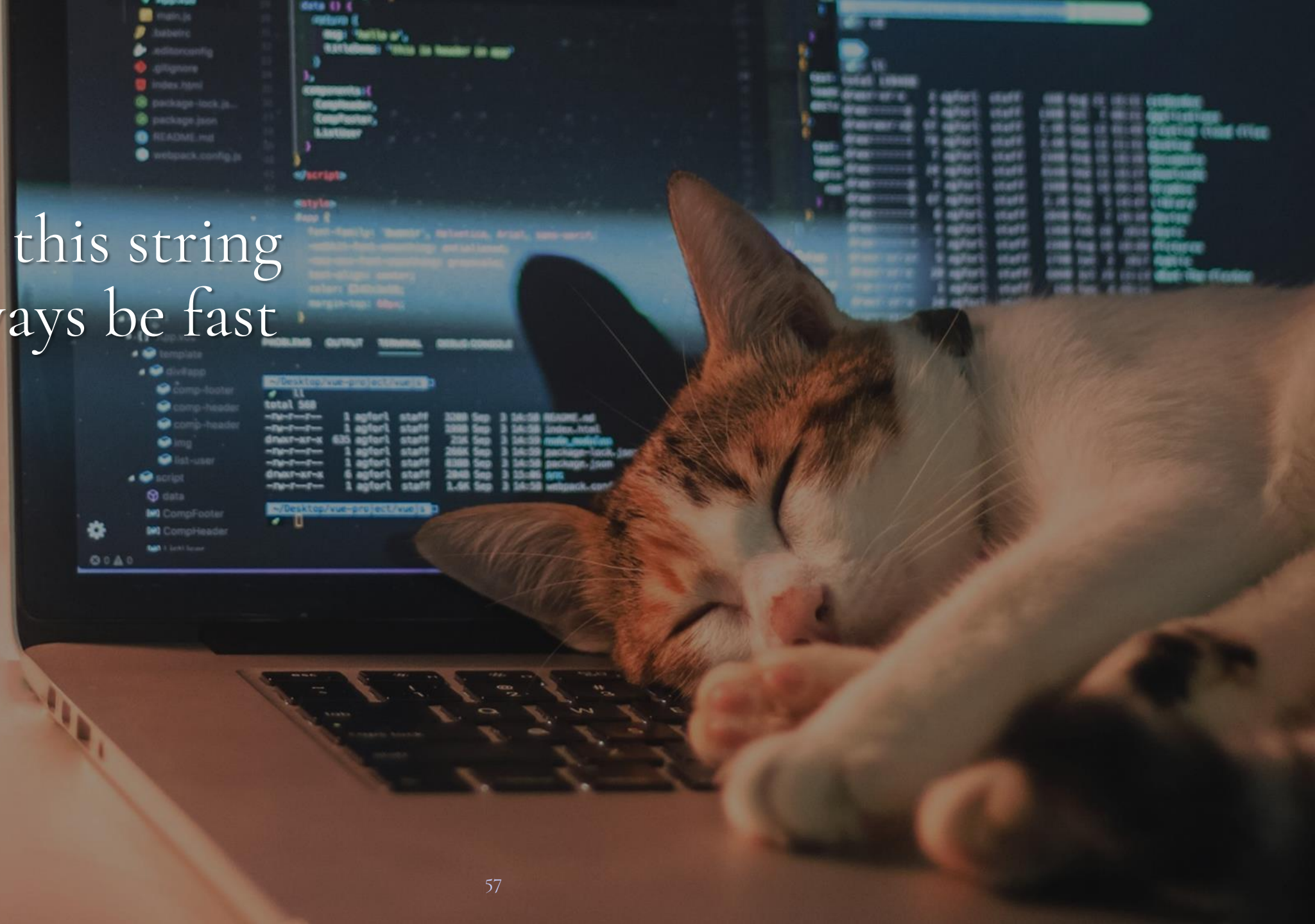


Like any experiment, we start with a hypothesis:  
our assumptions (mental models) about reality.

We can target our “this will always be true” assumptions that exist all over our stack.




Parsing this string  
will always be fast



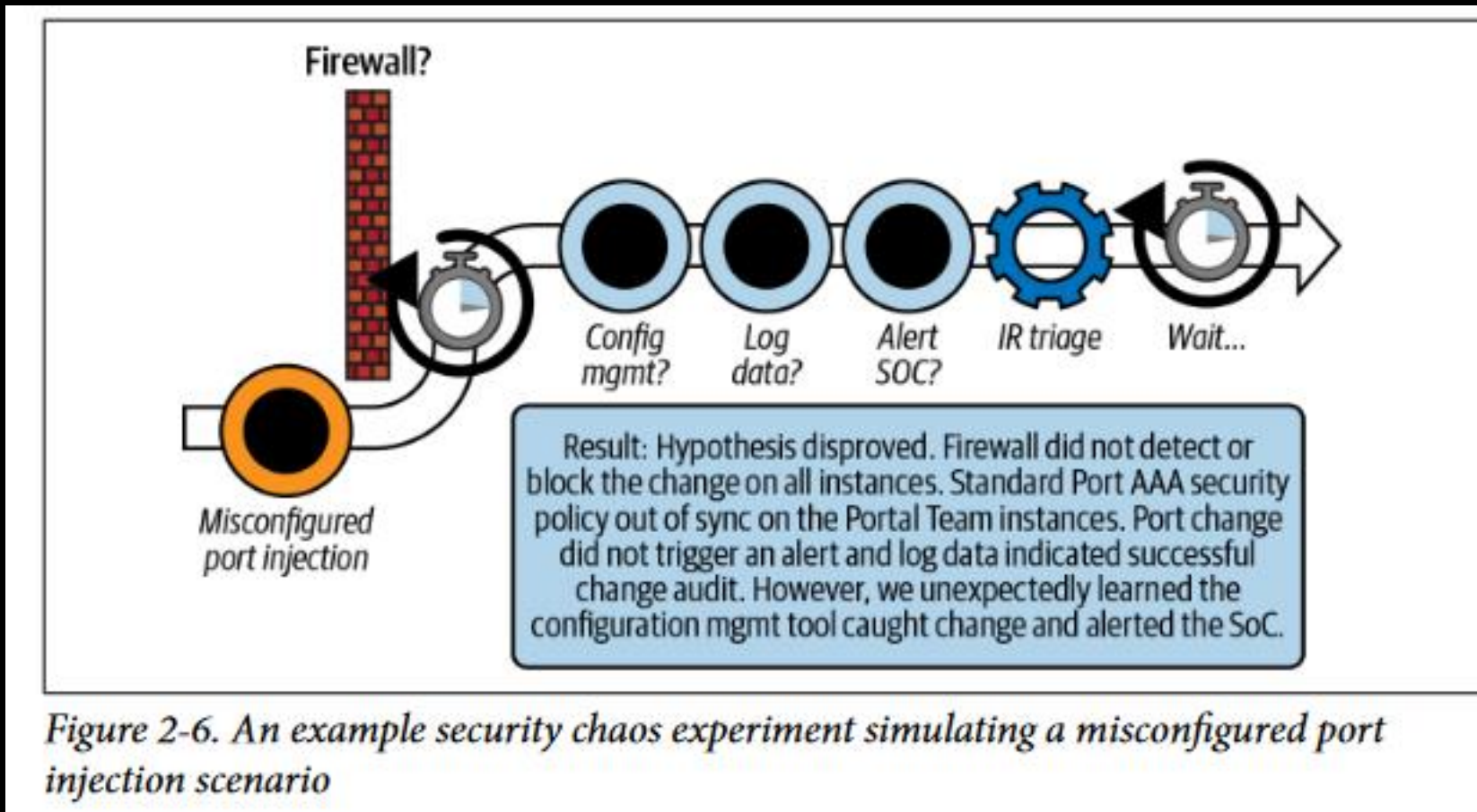


Messages on this  
port will always be  
post-authentication

A small, fluffy brown dog with a white chest patch is wearing dark-rimmed glasses. The dog is looking down at a tablet computer that is propped up on a wooden surface. The background is a plain, light-colored wall.

An alert will always fire if a  
malicious executable appears

Example hypothesis: "If a user accidentally or maliciously introduced a misconfigured port, we would immediately detect, block, and alert on the event."



What other experiments are relevant to SREs?  
There's ample overlap with security.













Each exposes how our sociotechnical system behaves in an adverse scenario *end to end*.

When we reveal the resilience properties of our systems, how do we capture this knowledge?



# III. Refining Mental Models with Decision Trees

The question isn't "is the system resilient?"  
It's instead: "the resilience of what, to what?"

Decision trees are a visual representation of different events possible in a scenario.





# Decision Tree Walkthrough



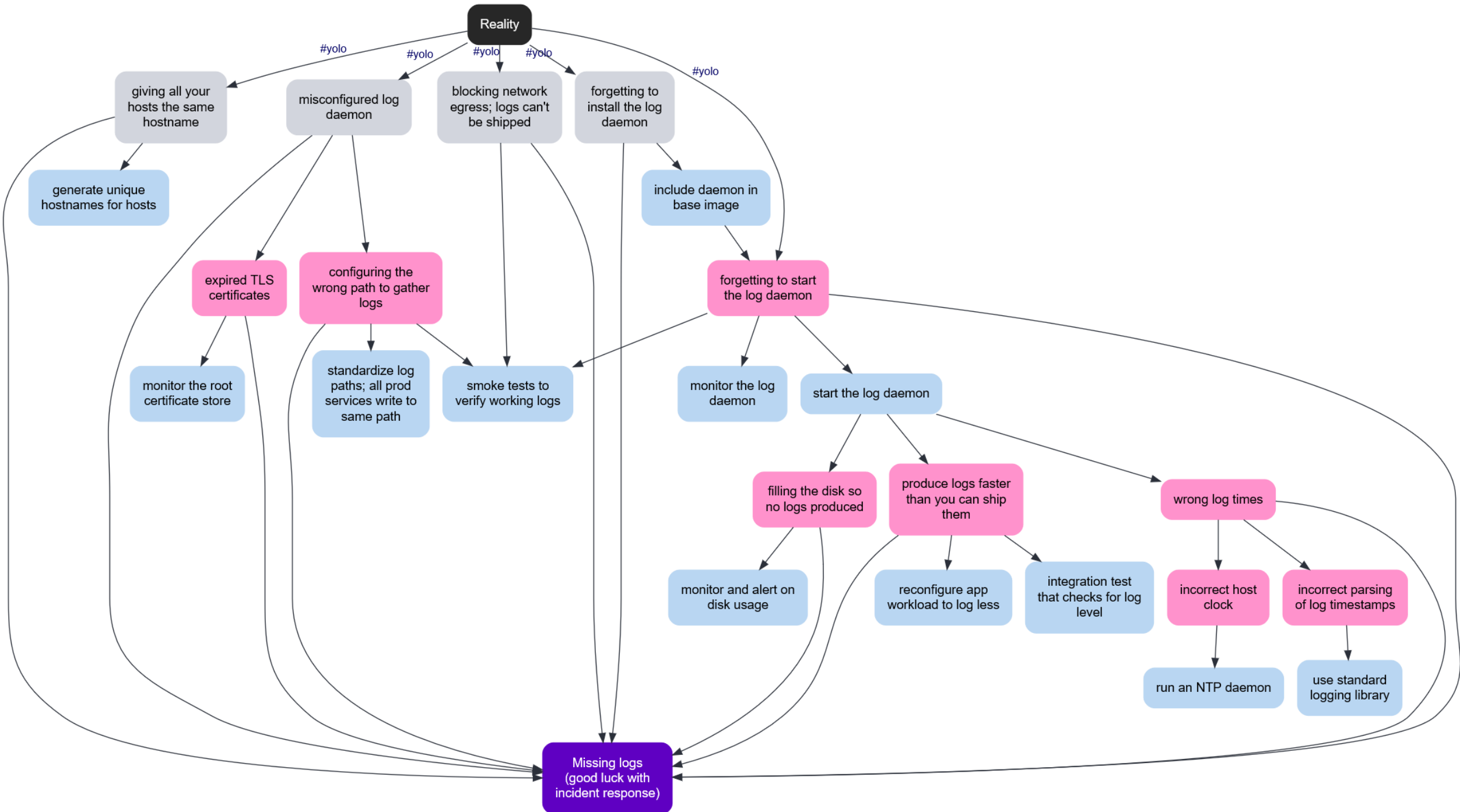
A close-up photograph of a flowering branch, likely from a tree or shrub. The branch is covered with numerous small, round, pink buds and a few larger, open flowers with light pink petals and yellow centers. The background is a soft, out-of-focus blue and green, suggesting a natural outdoor setting. The text is overlaid on the image in a white, sans-serif font.

In general, decision trees map how adverse events and mitigations unfold across spacetime.

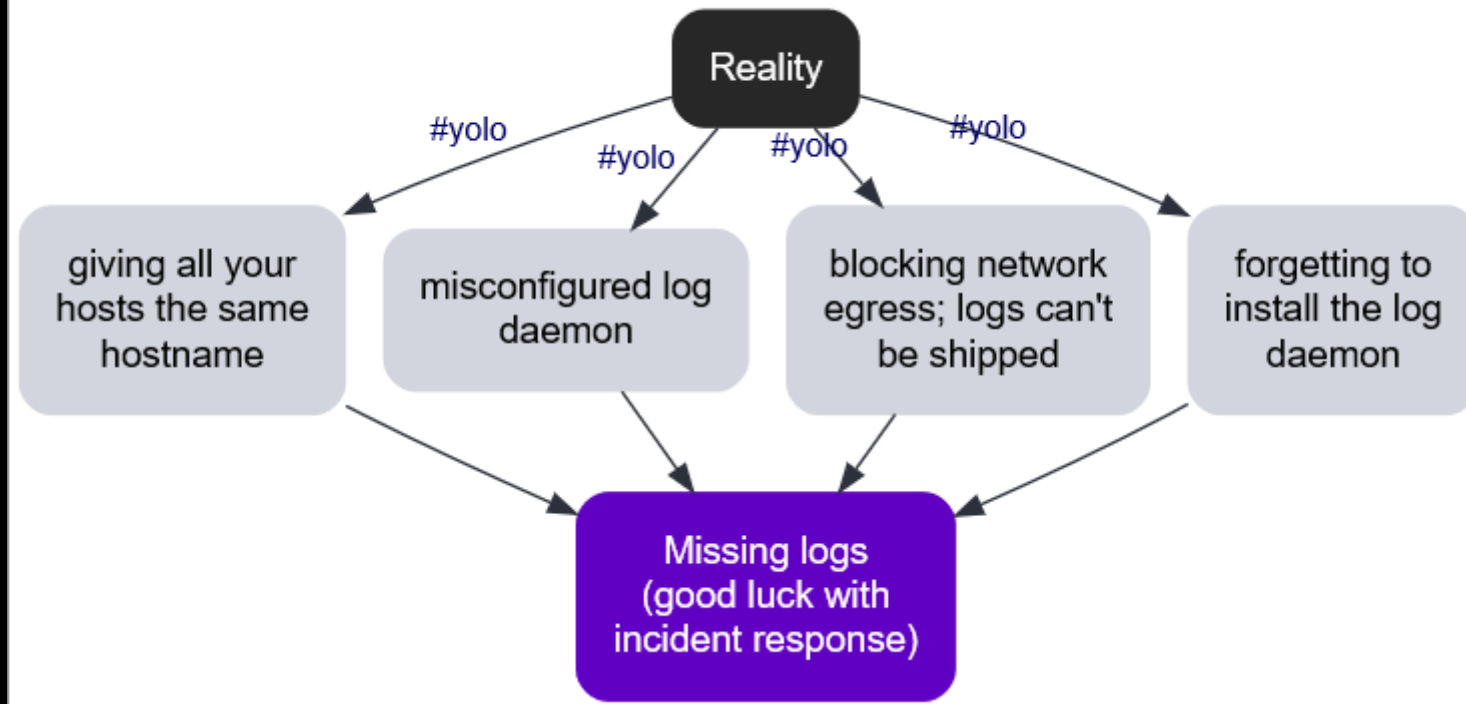
Security decision trees map attacker choices and visualize their paths through the system.

The point isn't perfection; it is iteration that keeps us honest about our mental models.

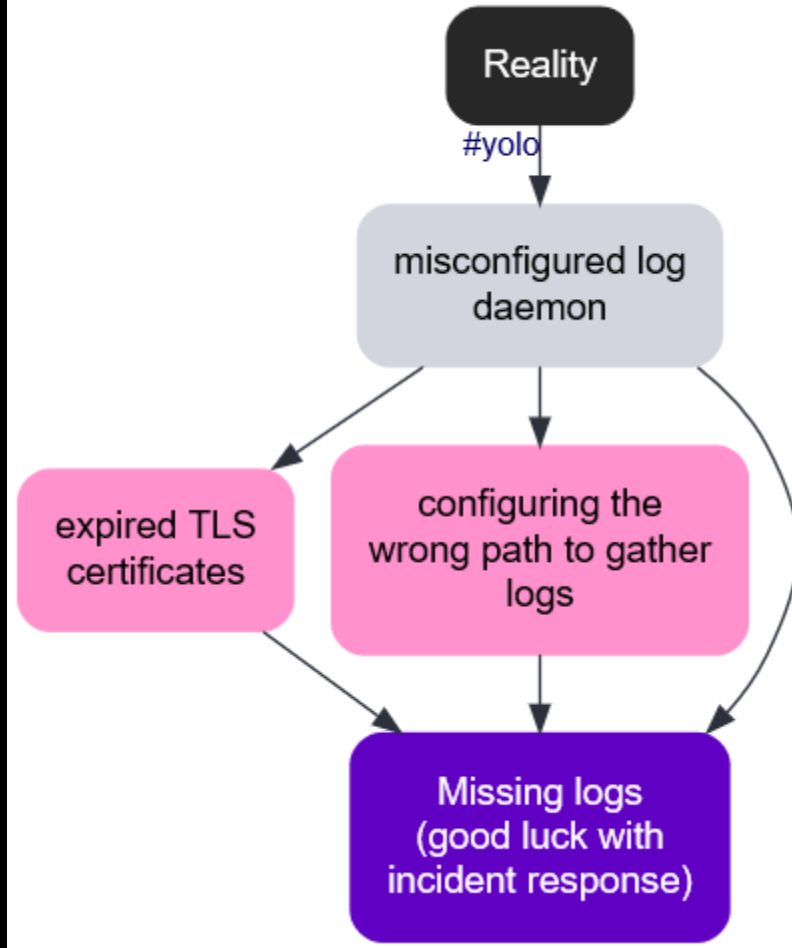
# Adverse Scenario - Missing Logs



## Adverse Scenario - Missing Logs

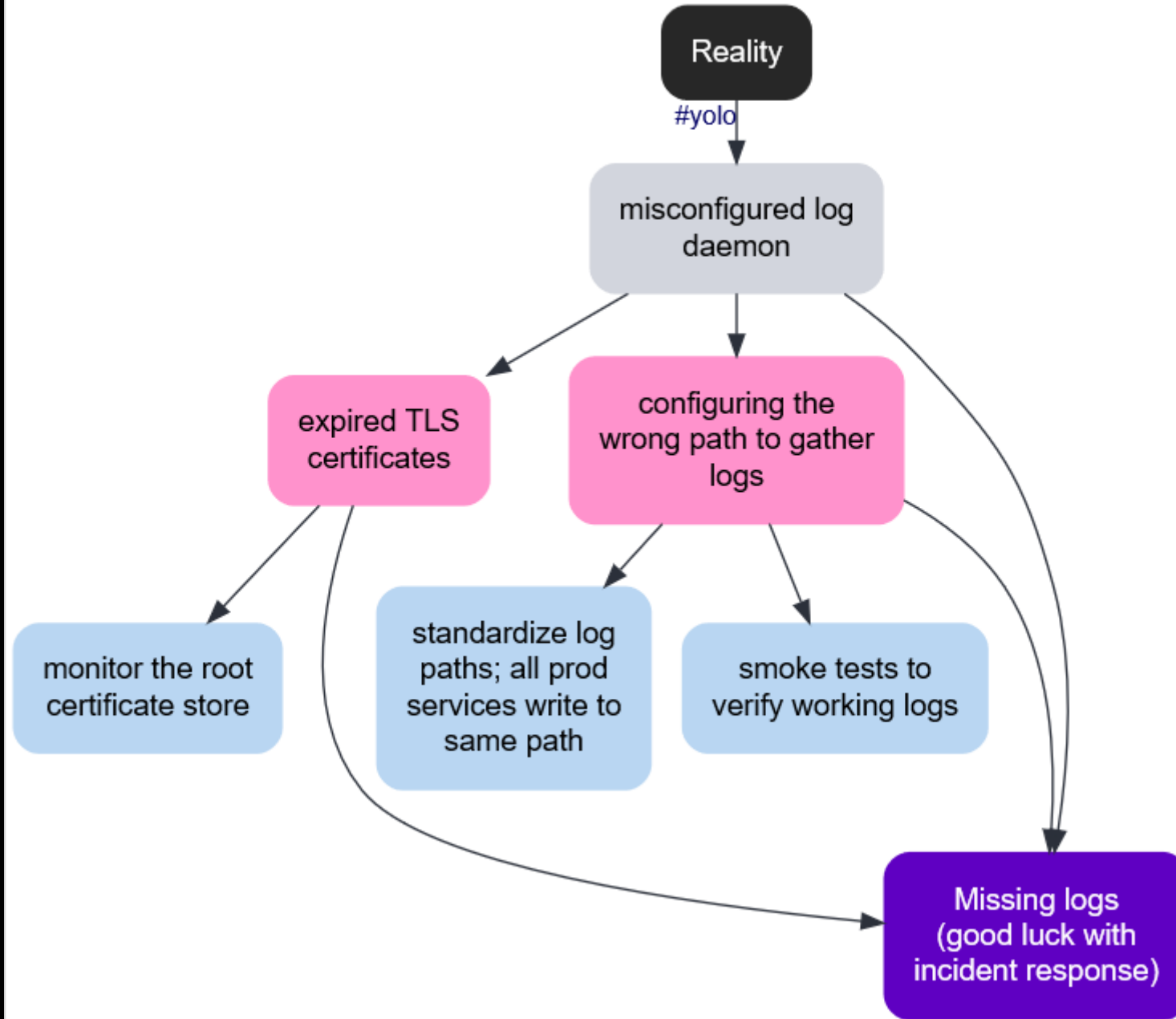


## Adverse Scenario - Missing Logs

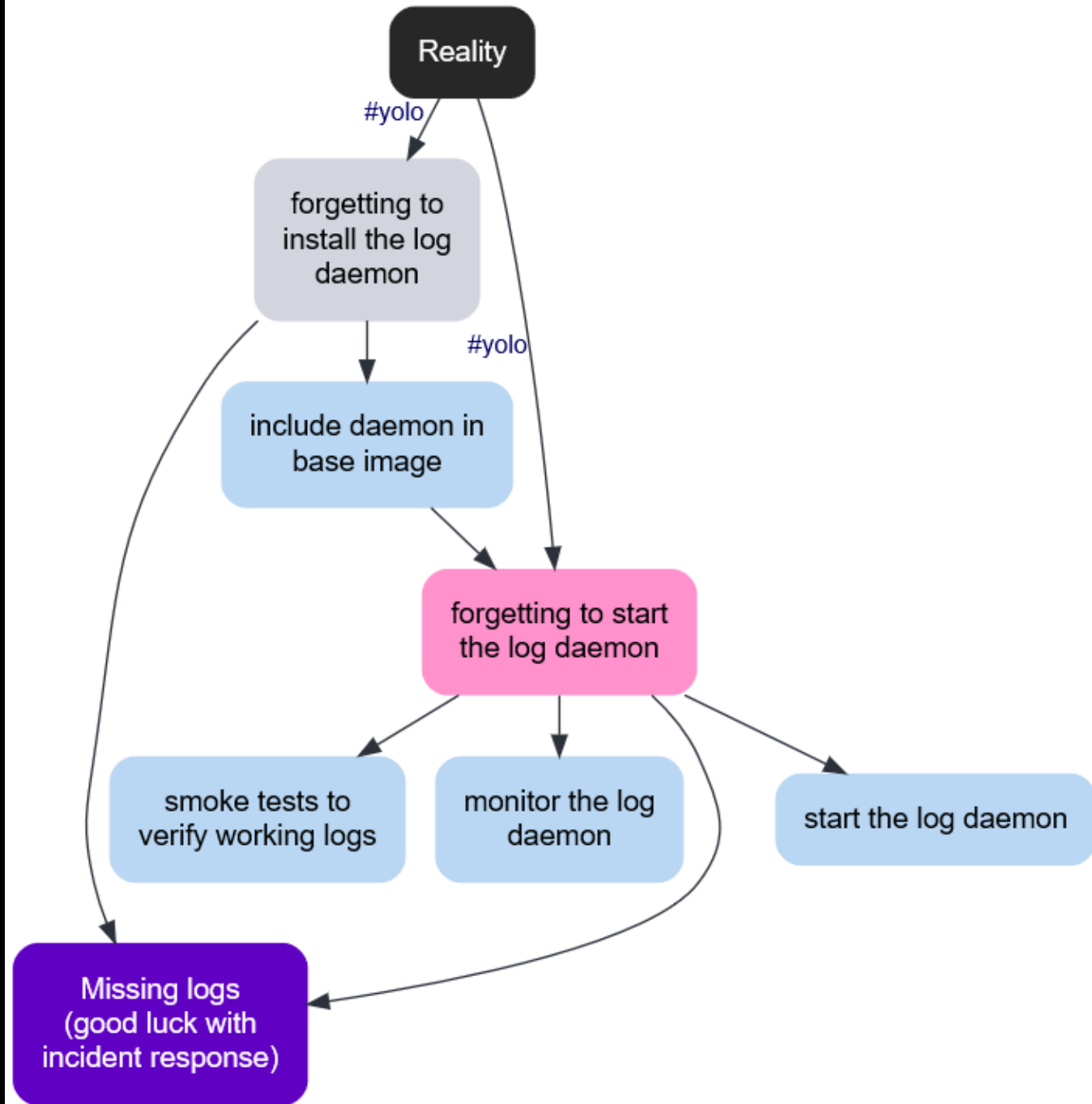




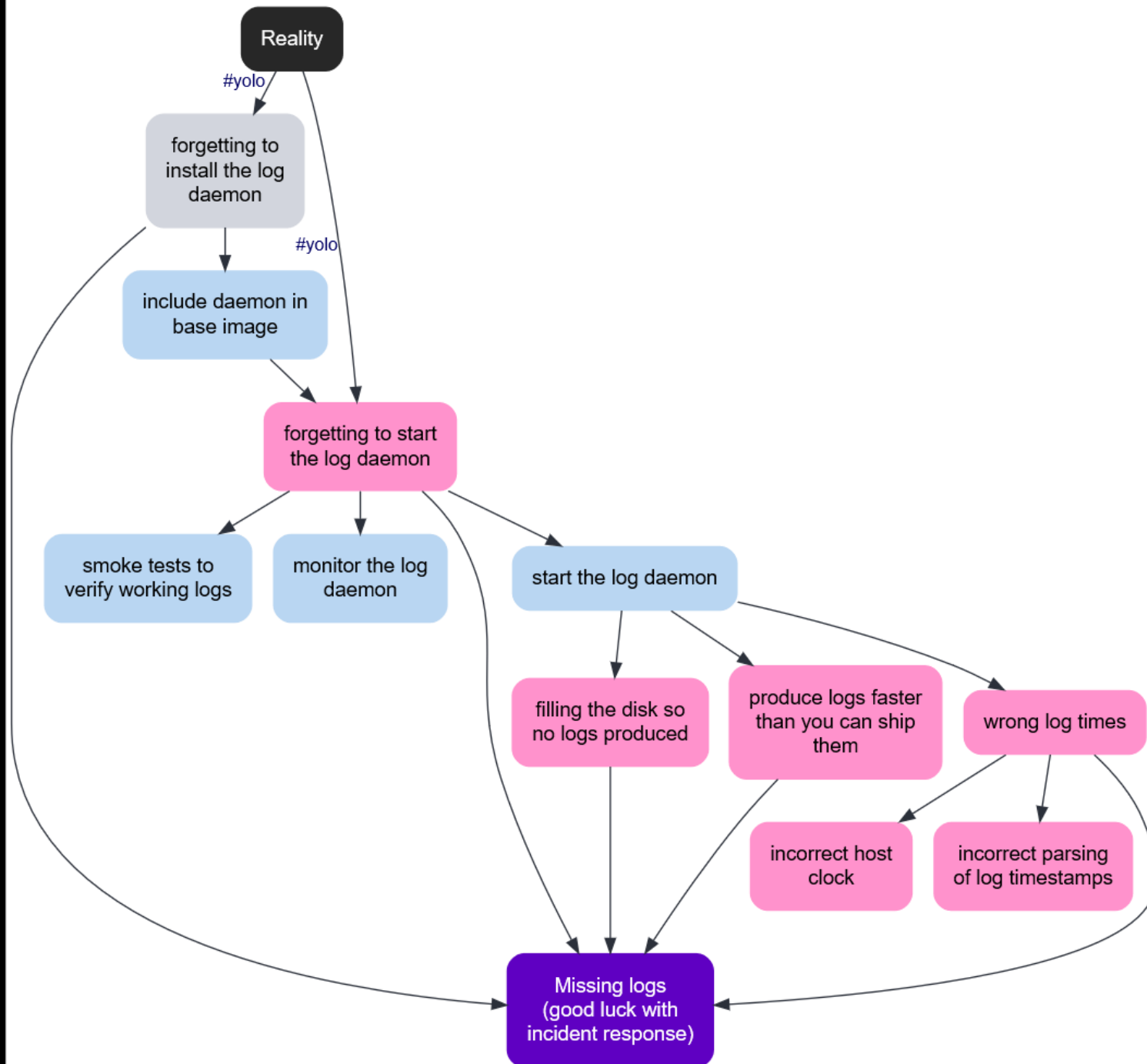
## Adverse Scenario - Missing Logs



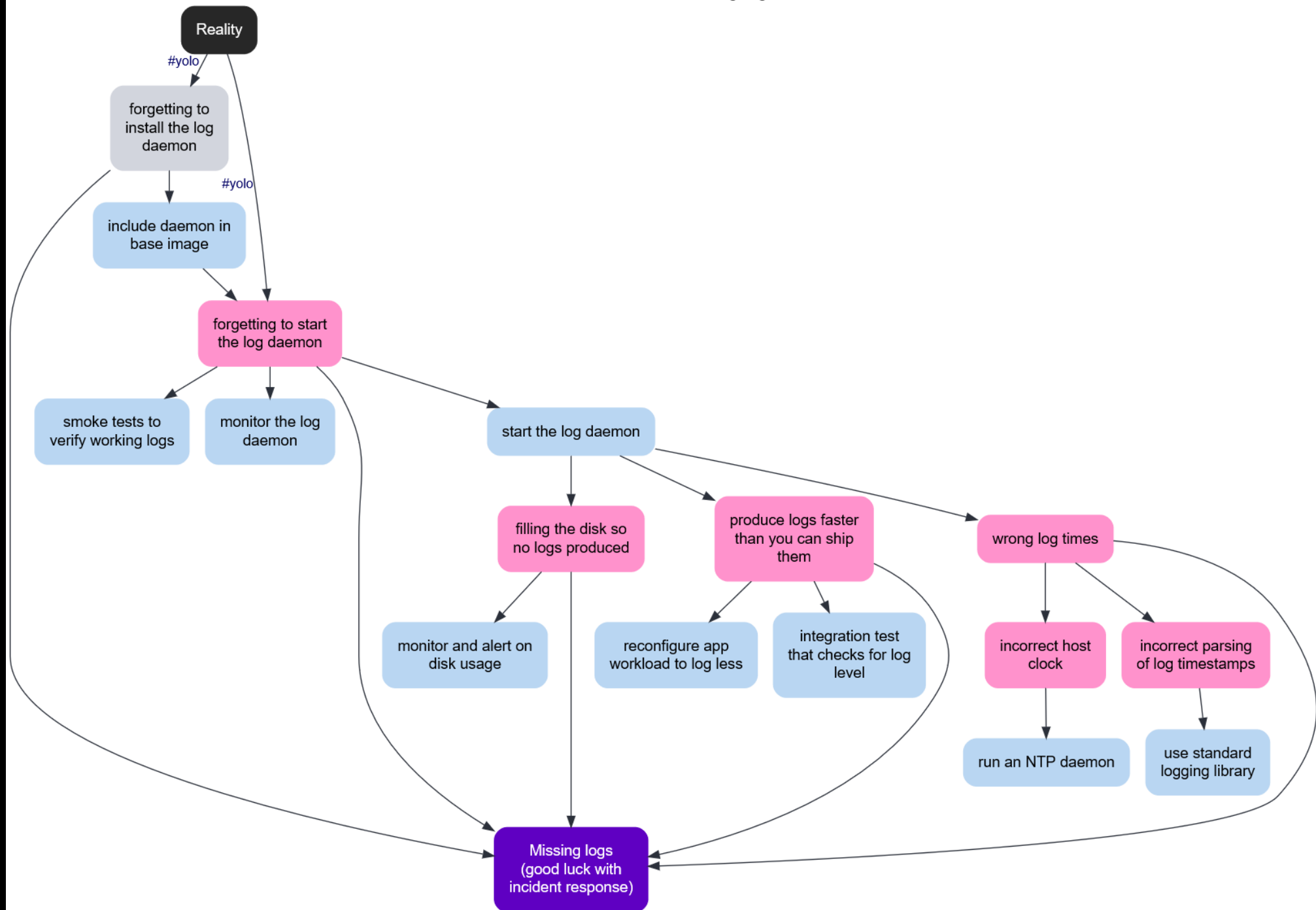
## Adverse Scenario - Missing Logs



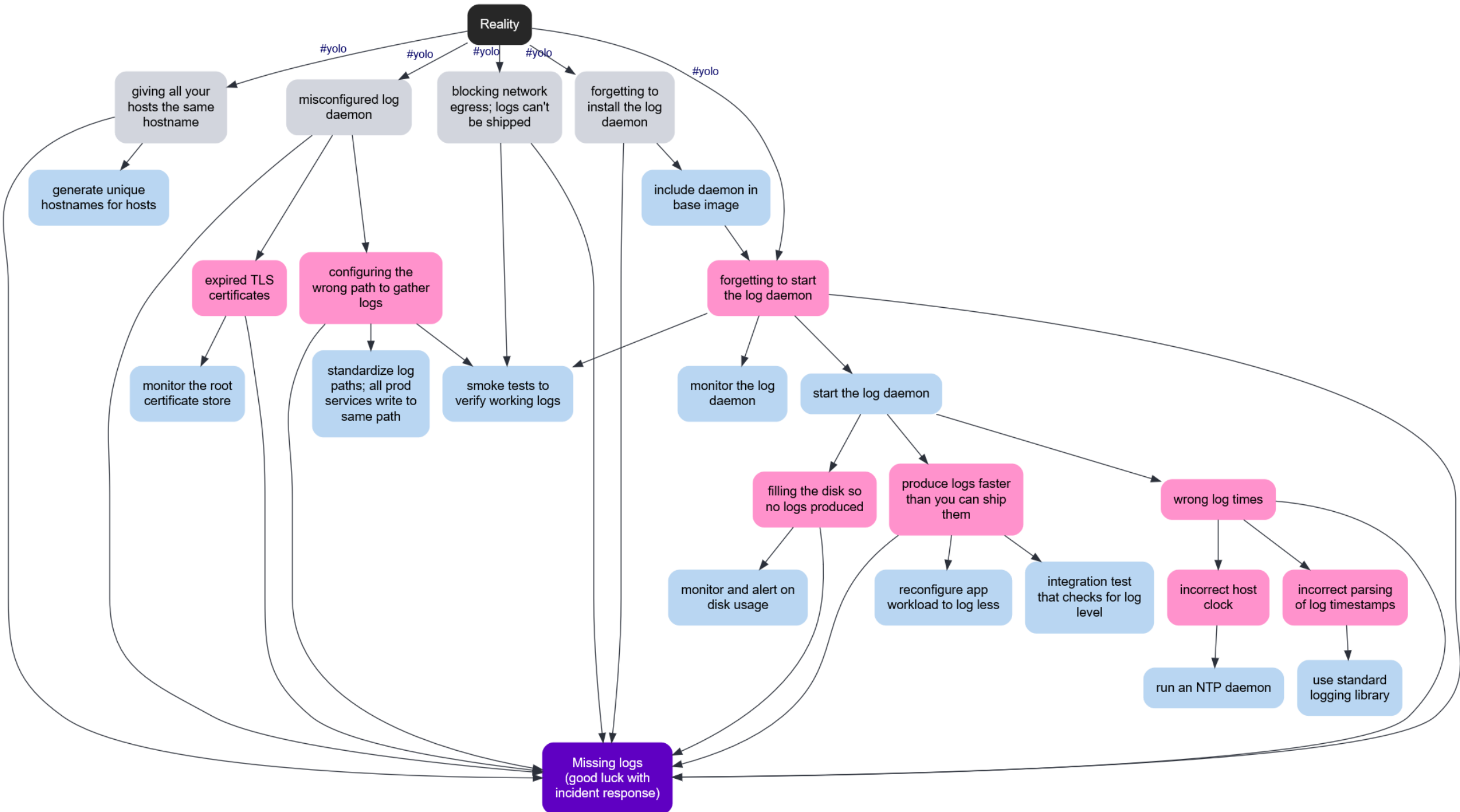
# Adverse Scenario - Missing Logs



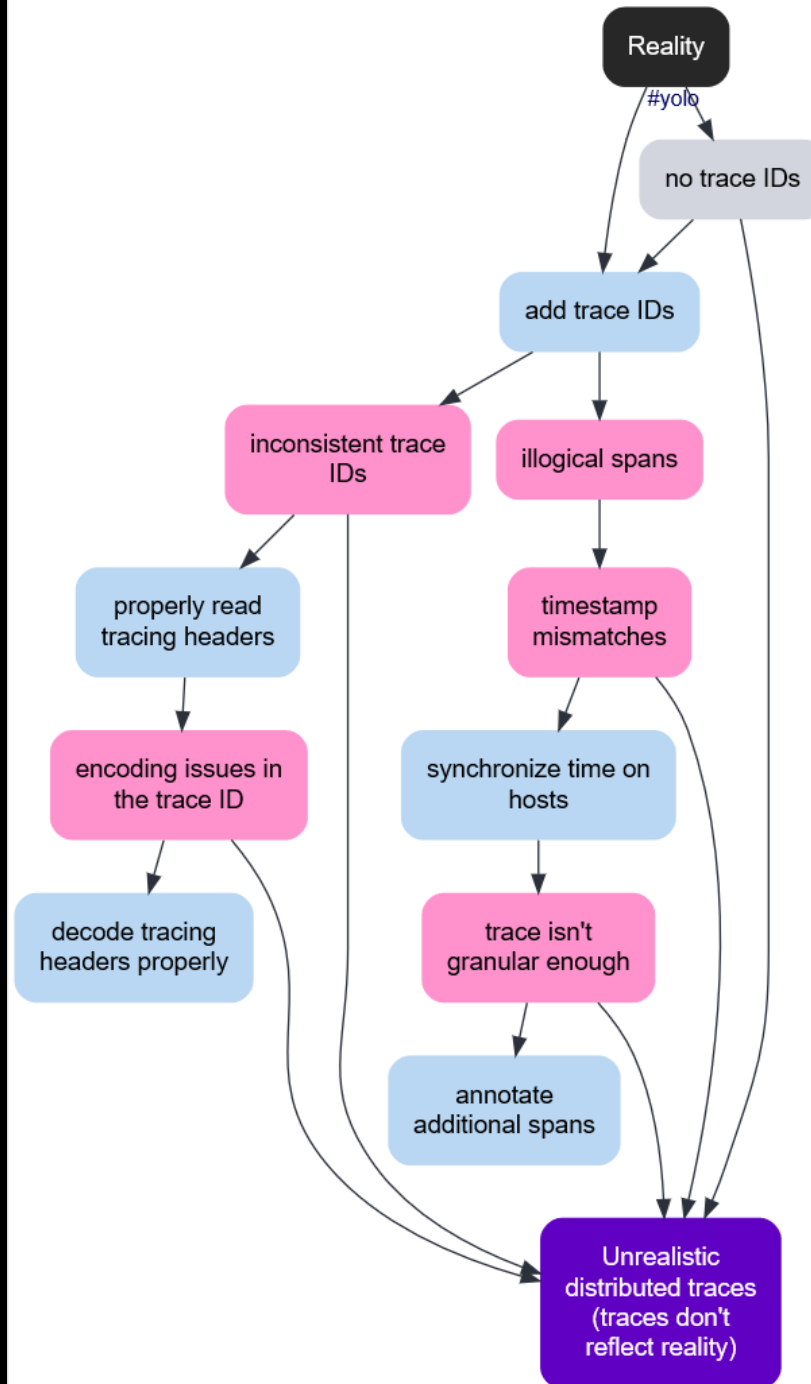
# Adverse Scenario - Missing Logs

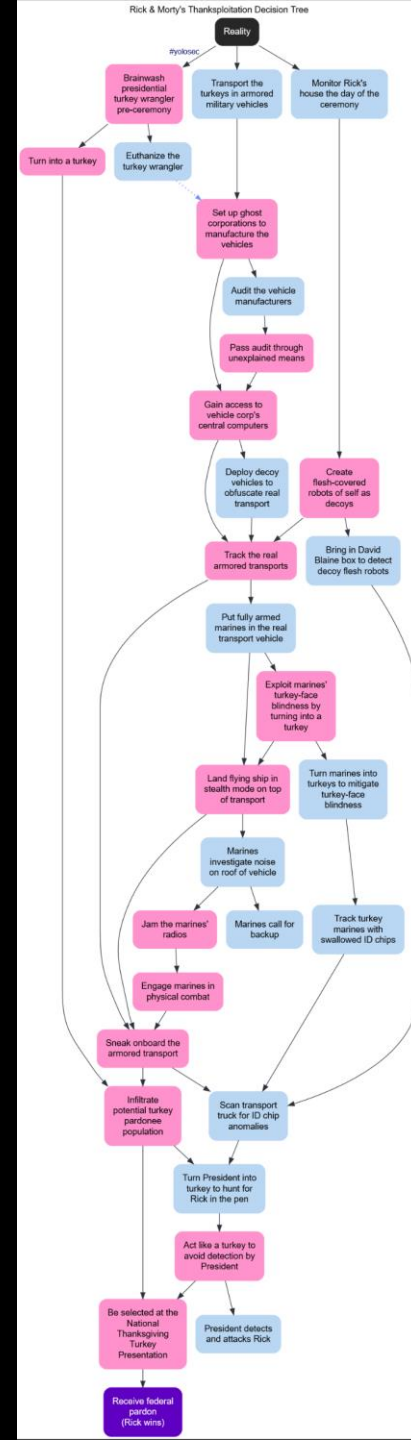


# Adverse Scenario - Missing Logs



# Adverse Scenario - Unrealistic Distributed Traces







# Applying Decision Trees in Practice



Decision trees cover all issues relevant in your system – including any “solved” ones.



Decision trees capture system architecture and flows, plus any gaps where we're unprepared.

Experiments form a feedback loop with decision trees to refine your mental models.



Decision trees help us continuously refine system architecture to sustain resilience.

They're also valuable during incident reviews — see where your assumptions held true or false.



Did events unfold in the flow modeled in the decision tree?

Did your mechanisms alter the flow or sequence of events as anticipated?

A small bird with a vibrant red head and neck, and a brown, streaked body, is perched on a dark branch. The bird is facing right and has a small pink flower in its beak. The background is a soft-focus blue sky with several branches of pink cherry blossoms in bloom. The overall scene is peaceful and natural.

What events or actions were missing from your assumptions?



Were there mitigations you didn't expect?

You can also document decision trees before your chaos experiments as hypotheses.

A close-up photograph of several pink dogwood flowers in bloom. The flowers have four large, light pink petals with darker pink veins and a white center. The centers are green and yellow. The background is a soft, out-of-focus greyish-brown.

We can also assess the potential efficacy of an architectural change through experiments.

Each design + experiment iteration refines  
decision trees and thus your mental models.

A close-up photograph of a branch with several light pink cherry blossoms. The flowers are in various stages of bloom, with some showing prominent stamens. The background is a soft, out-of-focus blur of more blossoms and branches, creating a shallow depth of field. The overall tone is gentle and natural.

Starting experimentation along the easiest  
branches verifies “obvious” assumptions

Once you gain confidence in resilience to obvious failures, you can move onwards



You may never get to the “super hard” failures branch when there are many ongoing changes

# IV. Conclusion







Our software systems are inherently socio-technical in nature; humans are how they adapt.



To adapt successfully in a complex system, we must continuously refine our mental models.



We can expose gaps in our understanding of the system's reality through chaos experiments.

A close-up photograph of pink cherry blossoms in full bloom, with dark brown branches visible. The background is a soft, out-of-focus field of more blossoms, creating a bokeh effect. The overall color palette is dominated by various shades of pink and light purple, with the dark branches providing a natural frame.

With this evidence in hand, we can capture our evolving knowledge in decision trees.

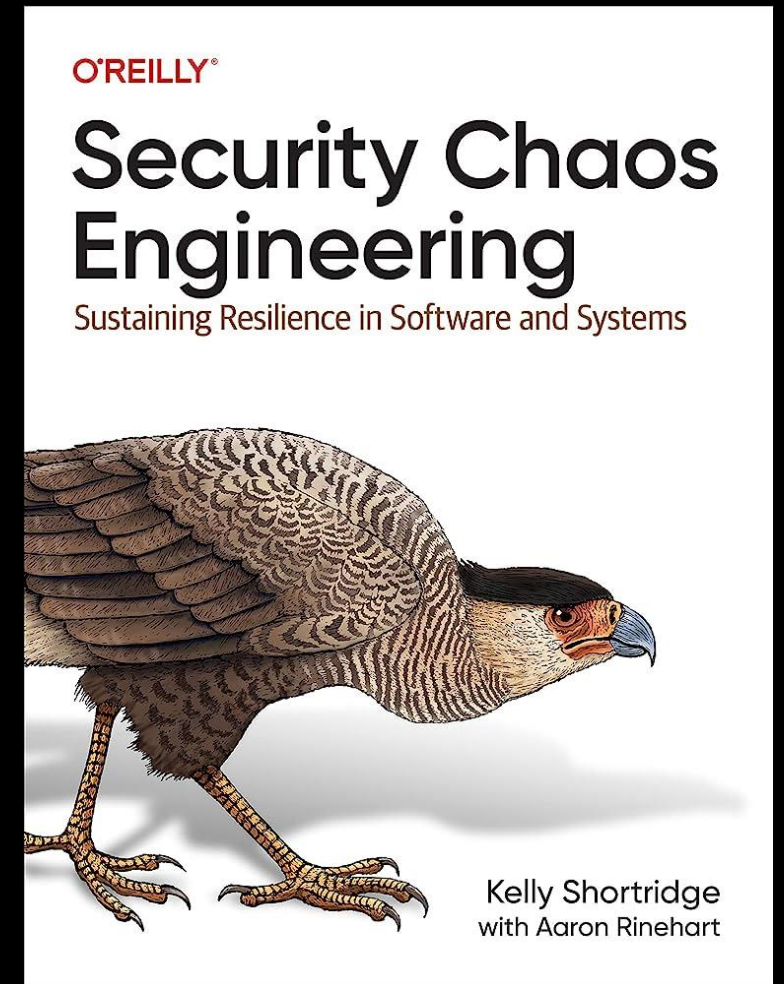
A lush tropical forest scene featuring a waterfall cascading into a stream. The stream flows over mossy rocks, surrounded by dense green foliage, including large ferns and trees. The lighting is soft and dappled, creating a serene and natural atmosphere.

From there, we can complete the feedback loop  
on which resilience and reliability depends.

Preorder the book & stay tuned for its release in late April (next month!):

[Amazon](#)

[Bookshop](#)





/in/kellyshortridge



@swagitda\_



shortridge@hachyderm.io



chat@shortridge.io