CSI Software Development: Using Blameless Postmortems for Learning and Growth

Santosh Hari

- Customer Engineer at Microsoft
- Extensive consulting and start-up experience
- Tech Community: Speaker, Ex-Azure MVP (2016-2020), Ex-organizer Orlando .NET User Group (onetug.net) & Orlando Codecamp (orlandocodecamp.com)

@santoshhari

@desinole

@santoshhari.dev

Bottom line my day job involves dealing with outages and the fallout and learnings
On a somewhat regular basis at some level
For example, middle of last year there was this issues with Crowdstrike
I've considered getting a t-shirt made saying "I survived Crowdstrike"
I've also had the opportunity of dealing with azure related outages
What I hope to do in this session is to show outages will happen
There is no way to prevent them
We can instead focus on learning from them
So we can respond better and reduce their impact
And improve our teams and organizations

Structure of talk

Horrible outage 1 and blame-filled fallout

Journey of learning and self-discovery due to a fake quote

Learnings on Outages, Postmortems & RCAs

Horrible outage 2 (spoiler alert: turns out much better)

Horrible Outage 1

The cast

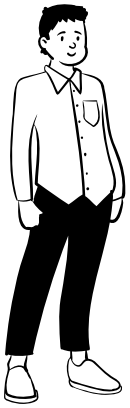
Tech Lead Taylor

VP Victor

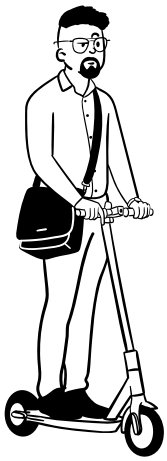
Manager Meena

The cast

Customer Carrie



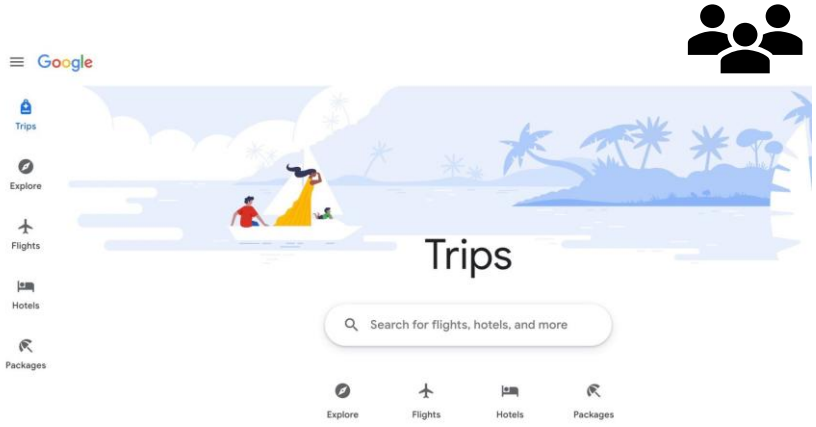
Ops Oliver



Developer Dave

Protagonist and product

Tech Lead Taylor



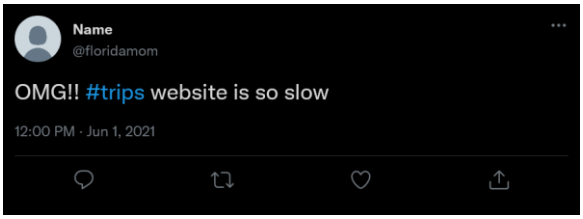
Taylor joined a successful start-up company named Trips (any resemblence is co-incidental) as Tech Lead
The company runs a SaaS solution for an industry vertical (think travel ticketing).

Customer flash sale ~8am

Customer Carrie

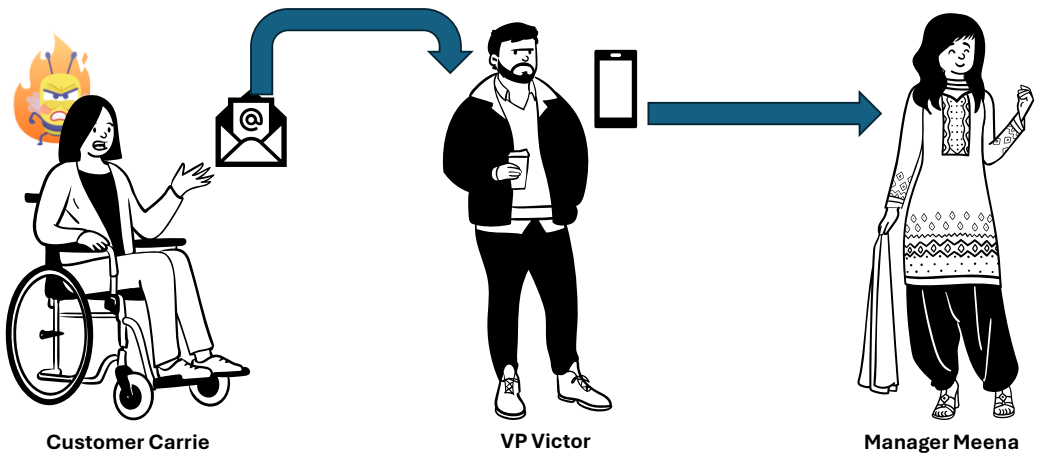
Please pay attention to the timeline at the top
On the Friday morning before a major weekend, when a lot of Trips' staff were out or just taking it easy and logging in late,
one of Trips' biggest customers customer carrie has a flash one day sale
Customer Carrie's campaign lands at 8am in user's mailbox
As a results users login to Trips' website en masse and start hammering the system

Website slowed down, users reacted ~8:40am

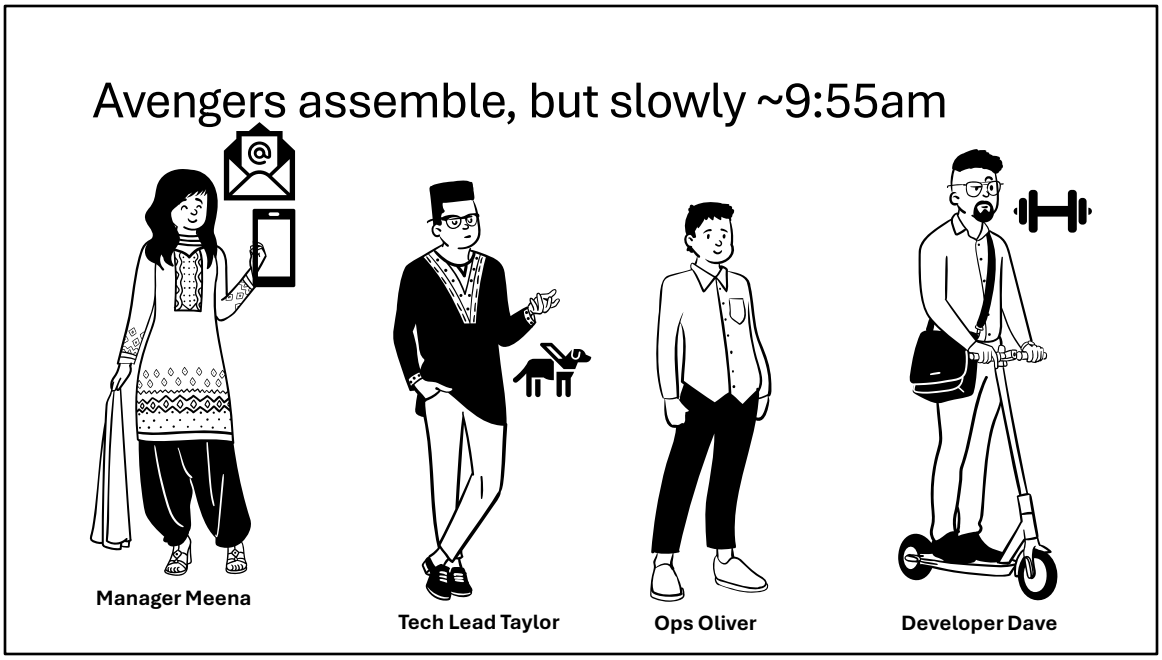


As sales heat up Trips' customers saw pages loading slower and slower
Being typical modern day customer of course they took to social media and started yelling at customer carrie's company

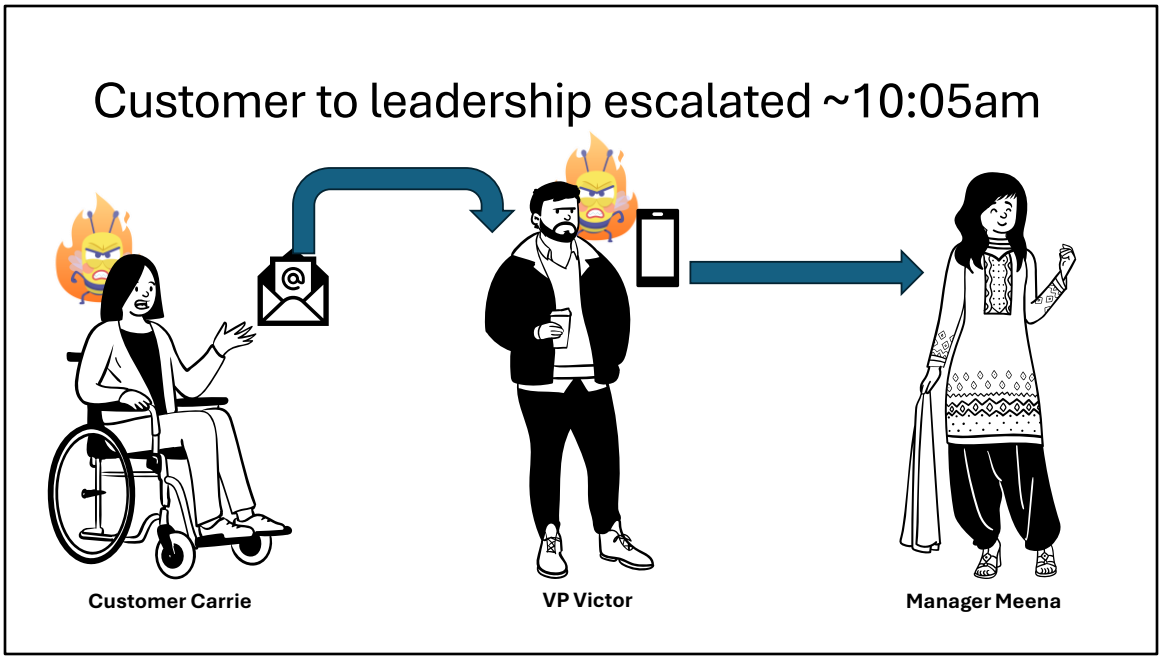
Trips heard from angry customer ~9:15am



Customer Carrie soon started hearing rumblings about the website issues from users on social media
And immediately fired off an angry email to Trips VP Victor
VP Victor logged in and tested the website and sure enough it was crawling
Victor immediately called Manager Meena in panic and asked to assemble the team

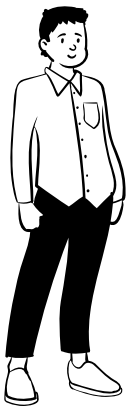


Manager Meena reached out to the team
Multiple frantic calls and emails went out
Tech Lead Taylor was out walking the dog
Developer Dave was at the gym
Ops Oliver was supposed to be off but thankfully responded and started looking on the infra side
It takes a while for the team to hop on a hastily assembled zoom bridge

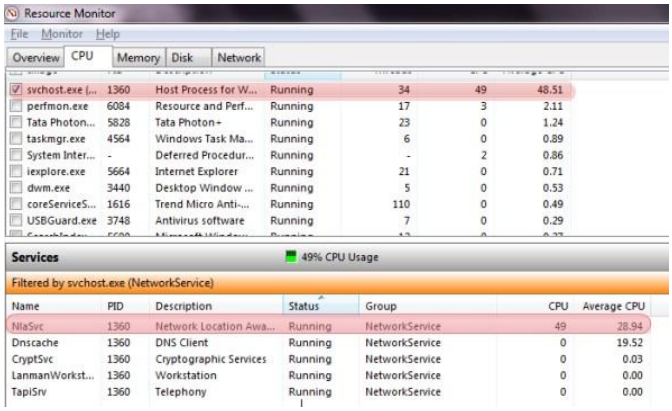


Meanwhile Customer Carrie was livid and repeatedly called VP Victor
Who provided assurances that the team was working on it
In turn VP Victor let Manager Meena have it about the slow response

Database server overloaded ~10:20am



Ops Oliver



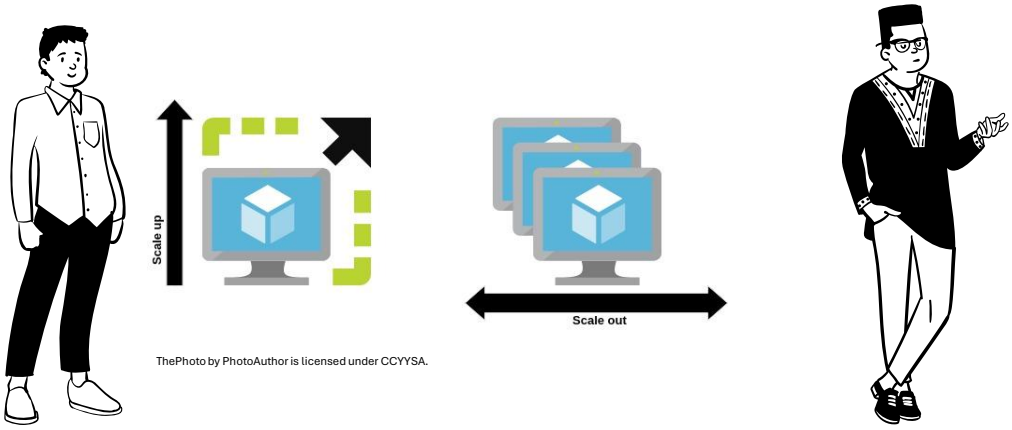
Tech Lead Taylor

Ops Oliver found underlying database hit limits which throttled query execution and caused timeouts and errors

Oliver reached out to Developer Dave and reported that something in the app was causing database load

Oliver also had to reach out Manager Meena and VP Victor to receive approval to scale up the database

Database server scaled up ~10:50am

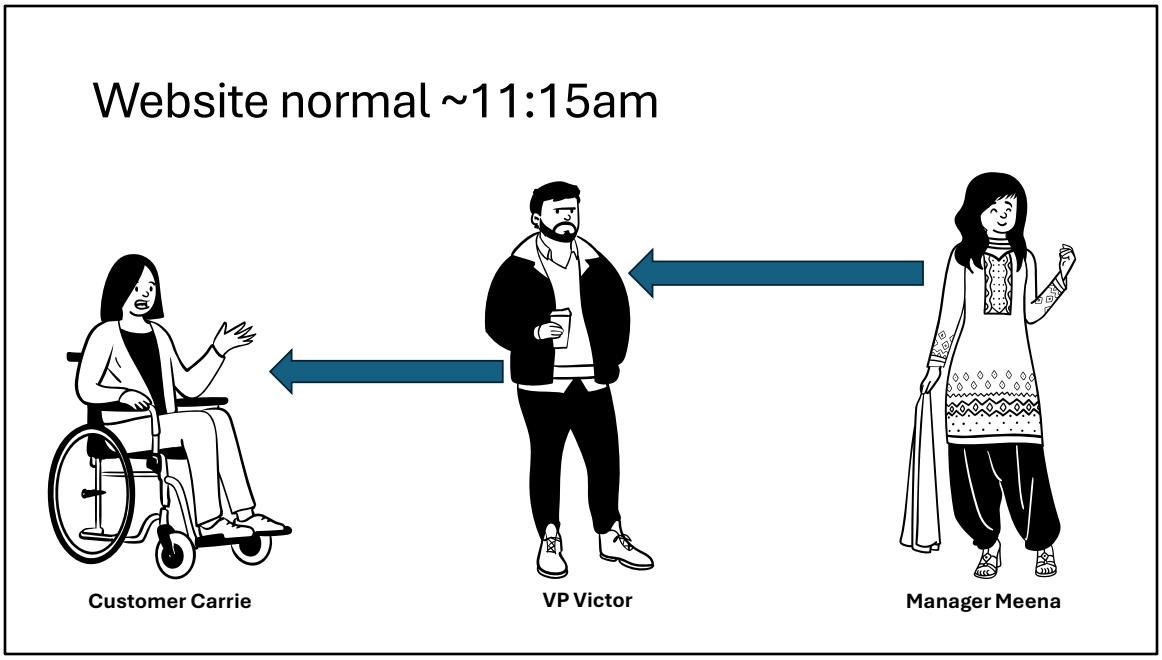


Tech Lead Taylor and Developer Dave dove into the codebase and started looking to see what could cause slow database queries

Meanwhile, Ops Oliver received approval from VP Victor to scale up to any level to fix the issue

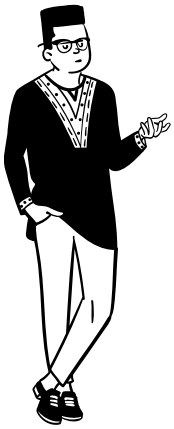
Ops Oliver initiated the scaling up process

Because Oliver did not know how much, basically double the DTUs on the Azure SQL database

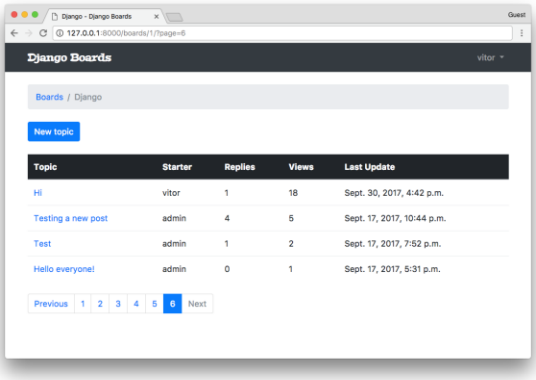


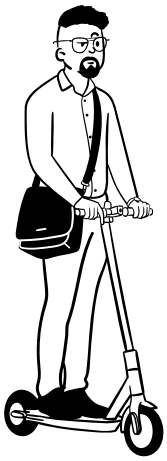
Ops Oliver noticed the database server load decreasing
Manager Meena immediately tested the website and found it responsive
Meena contacted VP Victor who passed on the happy news to Customer Carrie
Customer Carrier while relieved wanted a full explanation for what happened
VP Victor promised to provide details and make sure this never happened

Problem code identified ~1:30pm



Tech Lead Taylor

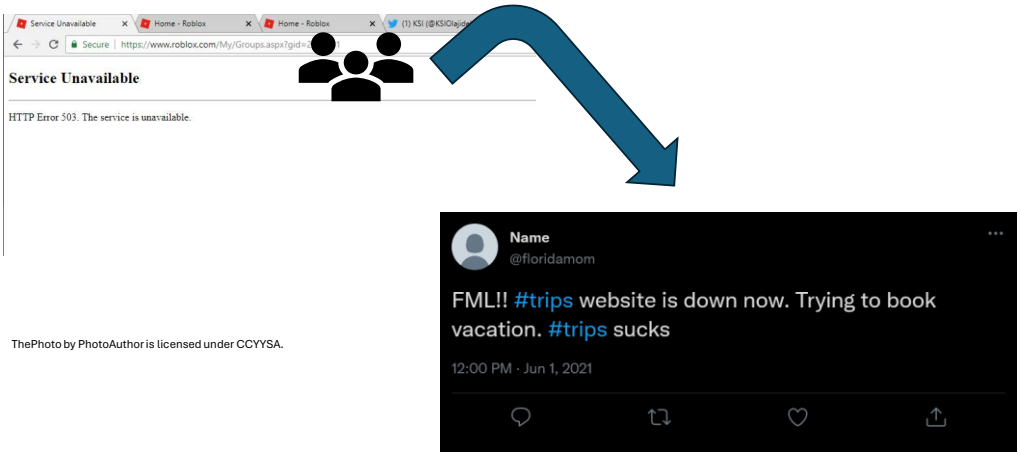




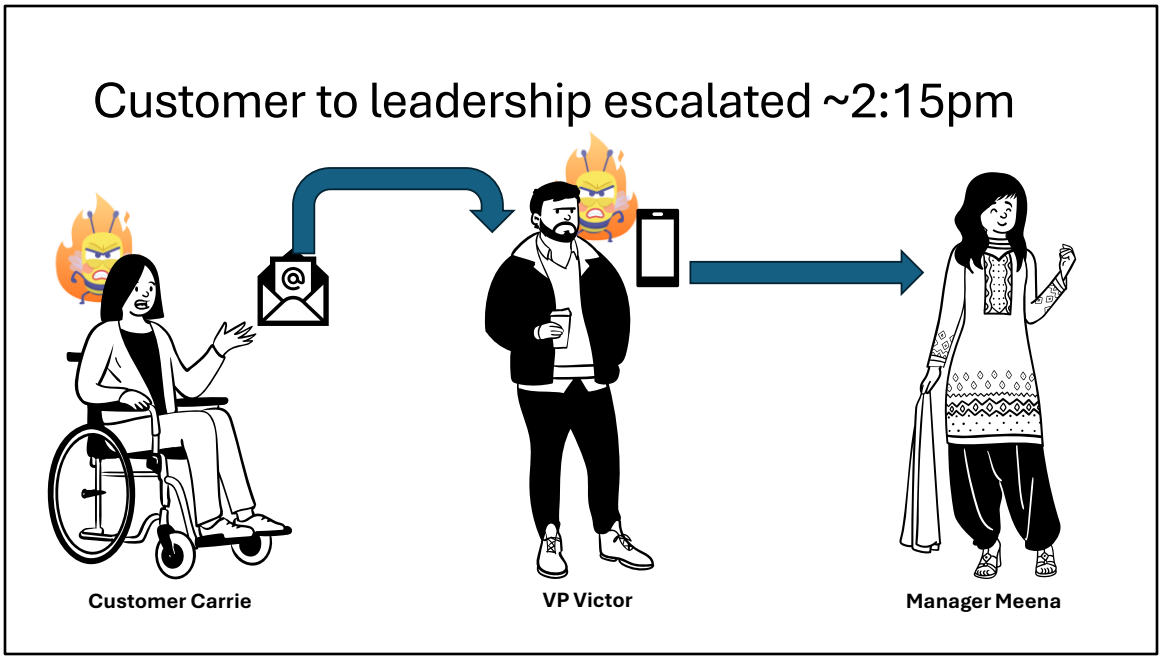
Developer Dave

Meanwhile Developer Dave was working hard to figure out what was causing database load
some recent code was added that fetched large datasets without pagination.
This was not noticeable under normal circumstances
but spike in traffic causes the underlying database to hit limits
which appeared to throttle query execution and caused timeouts and errors
Tech Lead Taylor put in a high priority work item to address this asap

Users started seeing intermittent "service unavailable" error ~2:00pm

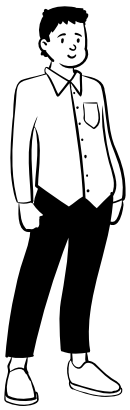


Scaling up seemed to relieve the issue, at least temporarily.
But soon Customer Carrie started receiving reports of users seeing intermittent 503 server errors.
This seemed to last for a few minutes, disappear and then rinse and repeat.

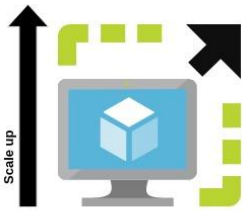


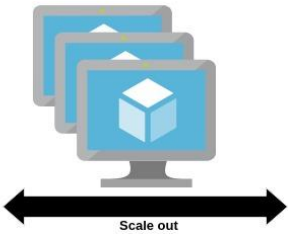
Thus started another escalation from Customer Carries to VP Victor
Who provided assurances that the team was working on it
In turn VP Victor let Manager Meena have it about the inability to handle issues

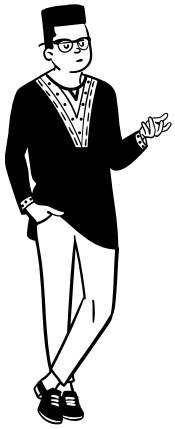
Database server scaled up ~2:20pm, no luck



Ops Oliver



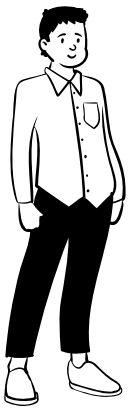




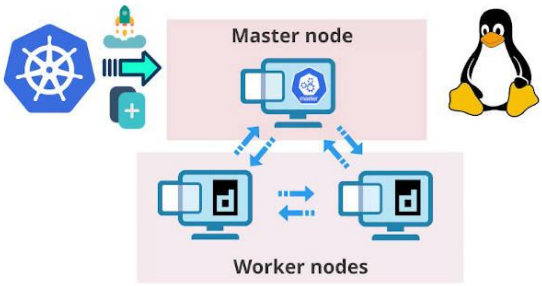
Tech Lead Taylor

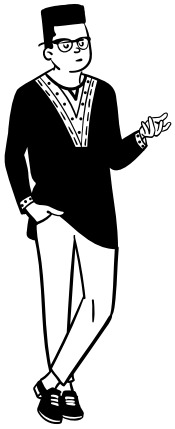
Tech Lead Taylor and Developer Dave correctly pointed out that this was a server side issue not a code issue
Ops Oliver initiated another database scaling up process
Because Oliver did not know how much, basically double the DTUs on the Azure SQL database but this did not help at all

App Server scaled out ~3:00pm



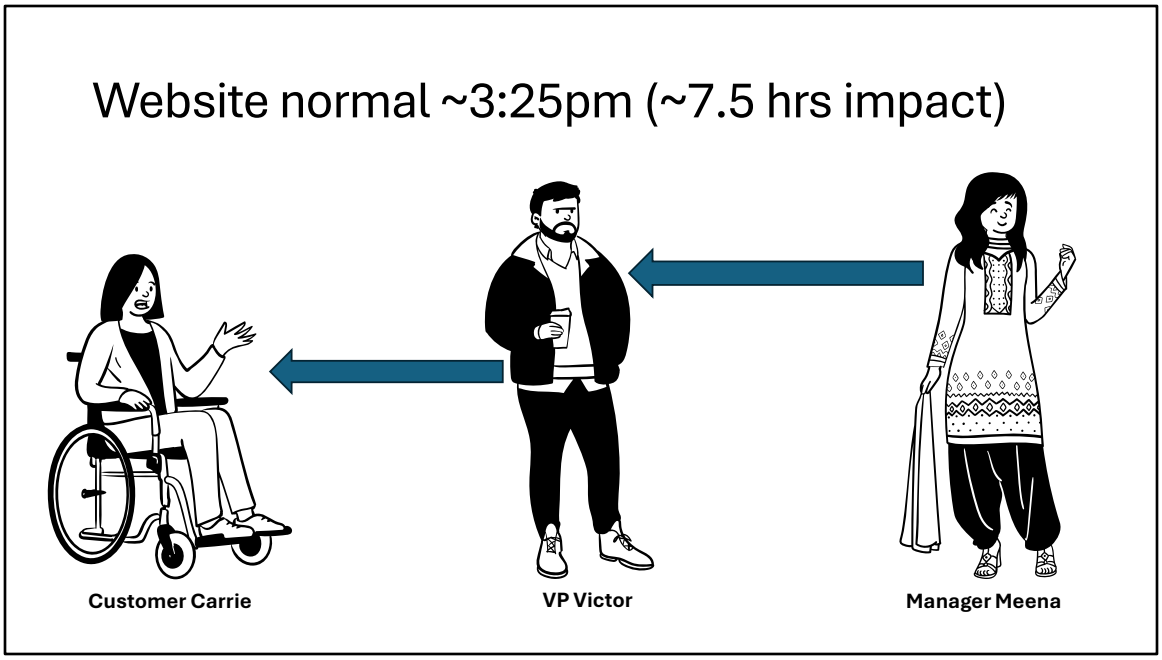
Ops Oliver





Tech Lead Taylor

The intermittent 503s continue.
On a hunch, Ops Oliver scaled out the app server
The worker nodes were doubled
This seemed to do the trick, after a while the 503s no longer showed up



Manager Meena immediately tested the website and found it responsive
Meena contacted VP Victor who passed on the happy news to Customer Carrie
Customer Carrier while relieved wanted a full explanation for what happened
VP Victor profusely apologizes to Customer Carrie and promises to look into it
Thankfully no more issues pop up during the remainder of the sale
Everyone wishes everyone else a good weekend with follow up calls scheduled for Monday

Enter the blame game

Monday a.m. feedback (Customer Carrie)

Customer Carrie

Issues

Financial loss

Lost confidence in Trips platform due to multi-hour outage

Why did we hear from customers instead of Trips staff

No ETA provided

Solution

Restructure contract with discount

Need website to stay up 24/7/365 with no downtimes and super-fast performance

Trips staff needs to work on staying on top of their platform health and inform Customer Carrie

15-minute updates with ETA

Monday a.m. follow up (VP Victor)

Issues

We lost face with Customer Carrie

Updates were infrequent and not meaningful (no ETA)

Solutions

Website should always be super-fast and not have downtimes

Manager Meena should provide 30-minute updates with ETA

VP Victor

Monday a.m. follow up (Manager Meena)

Issues

- Had trouble finding people during crisis
- Updates were infrequent and meaningful (no ETA)

Solutions

- Every team member should inform Manager if away from desk and have access to email/phone when away
- Tech Lead Taylor should provide updates every 30 minutes with ETA

Manager Meena

Monday a.m. follow up (Tech Lead Taylor)

Issues

- Last-minute code change triggered the issue
- Database and app server scaling faulty

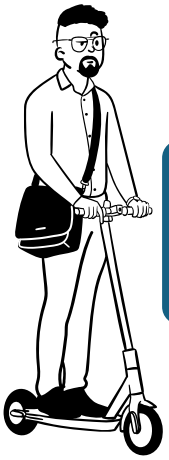
Solutions

- Additional approvals for code check-ins
- Provision double estimated resources for safety

Tech Lead Taylor

- Issues
  - Last-minute code change triggered the issue
  - Database and app server scaling faulty
- Solutions
  - Additional approvals for code check-ins
  - Provision double estimated resources for safety

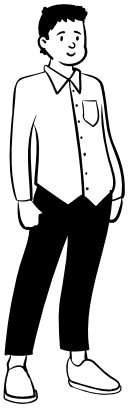
Technical battles



Developer Dave

Dave: "Clearly Ops did not have the scaling set correctly to serve additional traffic"

Oliver: "Dev needs to write better code. Also, management had Ops on a budget and we did the best we could"



Ops Oliver

Our tech lead also has to deal with finger pointing between Developer Dave and Ops Oliver

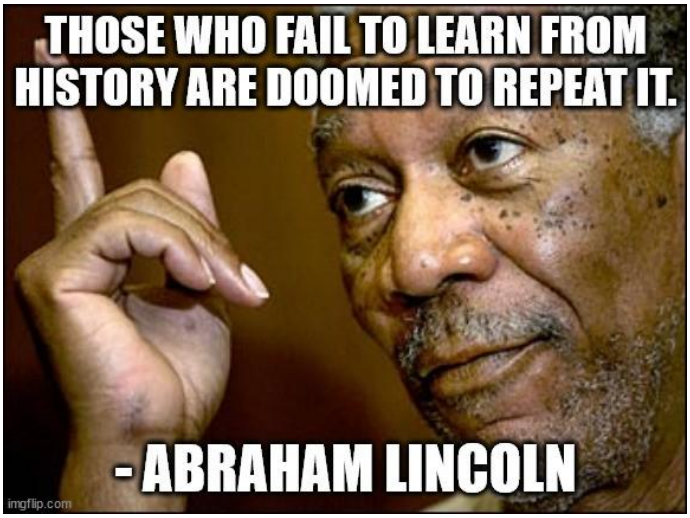
Dave: "Clearly Ops did not have the scaling set correctly to serve additional traffic"

Oliver: "Dev needs to write better code. Also management had me on a budget and I did the best I could"

Understanding Outages (Tech Lead Taylor perspective)

So we will come back to this story but now I want you to step into Tech Lead Taylor's shoes
Obviously most if not all the decisions made post-outage
Were not well thought out/reactionary
These cause major friction and loss of morale in the team
Subsequent outages lead to similar fallout and Tech Lead Taylor is at wit's end
Let's zoom in and pretend we're looking at things from Tech Lead Taylor's perspective

Journey inspired by (fake) inspirational LinkedIn quote



100% uptimes are impossible

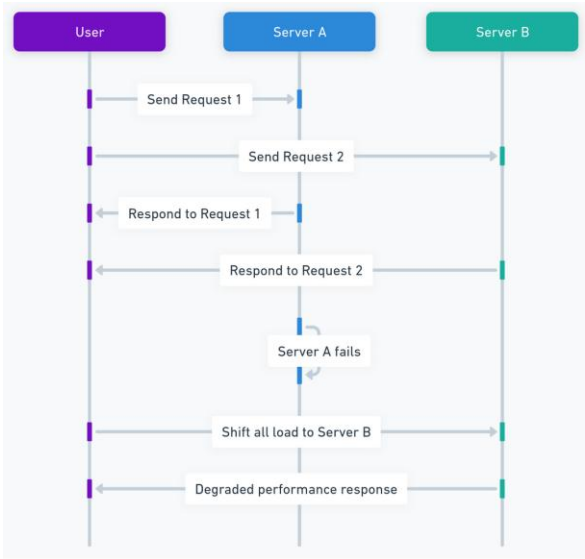
Uptime (%)	Downtime per Year	Downtime per Month	Downtime per Week	Downtime per Day
99.0	3.65 days	7.31 hours	1.68 hours	14.4 minutes
99.9	8.76 hours	43.83 minutes	10.1 minutes	1.44 minutes
99.99	52.56 minutes	4.38 minutes	1.01 minutes	8.64 seconds
99.999	5.26 minutes	26.3 seconds	6.05 seconds	0.864 seconds
99.9999	31.5 seconds	2.63 seconds	0.605 seconds	0.086 seconds

Taylor knows 100% uptime is not possible but is unable to convey this
The Taylor stumbles upon this matrix and the concept of SLA
Additionally Taylor also learns that every additional 9 increases the overall cost of the system and resources by nearly 10x
Aha Taylor thinks "I can use this to set customer and manager expectation on uptime"

Failures are by nature, unscheduled & inconvenient

You can't send a meeting invite in advance for an outage
Failures can happen at certain intervals,
But if you can't prevent them it does not help
Failures cause inconvenience to the the teams, customers, support personnel, and so on
I recommend reading State of DevOps report and Google SRE handbook for supporting evidence

Failures can cascade



You have probably seen or witnessed cascading failures at some point:
server A and Server B handle load in parallel.
Server A CPU usage suddenly spikes leading to Server A failure.
All the load shifts to Server B and cause server B to have degraded performance.

Due to system design and dependencies, these kind of scenarios are always in play

When an outage happens, watch out for cascades

Changes cause failure

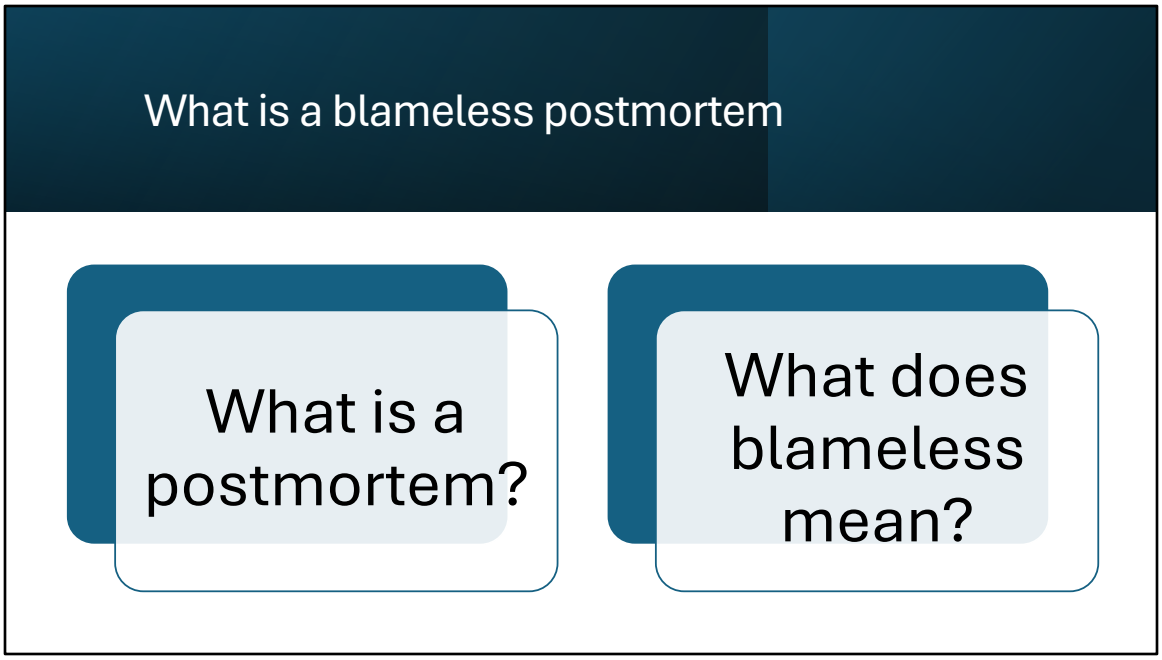
Performance level	Change lead time	Deployment frequency	Change fail rate	Failed deployment recovery time	Percentage of respondents*
Elite	Less than one day	On demand (multiple deploys per day)	5%	Less than one hour	19% (18-20%)
High	Between one day and one week	Between once per day and once per week	20%	Less than one day	22% (21-23%)
Medium	Between one week and one month	Between once per week and once per month	10%	Less than one day	35% (33-36%)
Low	Between one month and six months	Between once per month and once every six months	40%	Between one week and one month	25% (23-26%)

This table is directly from Accelerate State of DevOps report book
A significant portion of outages are caused by change
Google has different studies to where this number can be as high as 50% in some cases
Even the Elite of the Elite have 5% change fail rate.
There's a good chance 1 to 2 in 10 deployments cause some kind of failure in production
While pursuit of 0% failure rate is noble,
We still need to pursue venues like reducing impact, faster detection, faster recovery time
And above all learning from these failures

Outages are like a canary in a coal mine
When the canary stops singing there is a larger problem
Similarly when an outage happens there is a potential for a larger problem to exist
Are we simply going to take the canary out and bury it?
That is are we simply going to fix this issue and move on
Or are we going to look into why the canary stopped singing
i.e are we going to look for the larger problem

Postmortems & RCAs

Speaking of Learning and Collaboration
These are done through the process of incident postmortems and RCAs
In order to achieve this we need to not only understand
What happened but also why did it happen?
Is there a larger issue at play?
Are there some commonalities here?



What is a Postmortem?
A postmortem is a structured process conducted after an incident,
such as an outage or a critical failure, to analyze what happened, why it happened, and how to prevent similar incidents in the future.
It aims to transform a negative event into a learning opportunity.

A blameless approach to postmortems removes the focus
on assigning fault to individuals or teams and instead emphasizes understanding systemic factors.
The goal is to create a safe space where everyone can openly discuss what went wrong
without fear of punishment or retaliation.

Incident versus Problem

Incident

- Urgent
- What Happened?
- Requires immediate restoration of service
- Canary

Problem

- Systemic
- Why did it happen?
- Requires RCA
- Coal mine

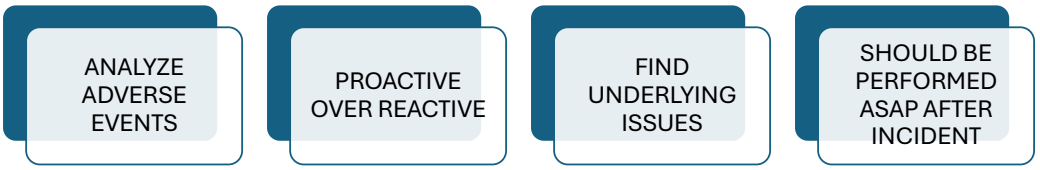
You will hear two terms incident management and problem management

incident management focuses on what happened,
whereas in problem management it focuses on the why.

incident management requires restoration at customer service,
problem management requires a root cause analysis.

From the previous example, overloaded database server
Scaling up was incident management
We never got a detailed explanation for why it happened and how to prevent it

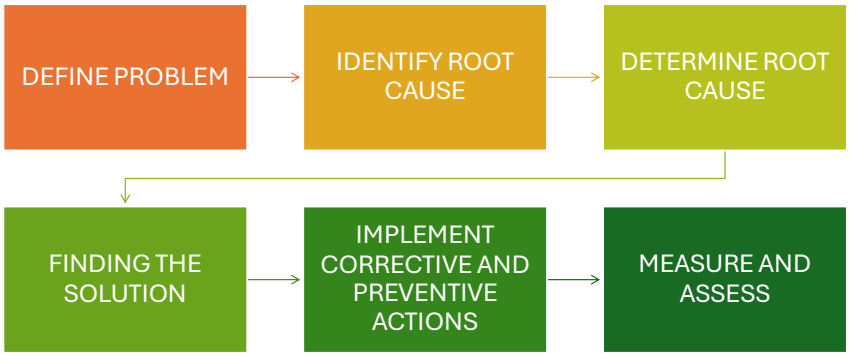
Root Cause Analysis (RCA)



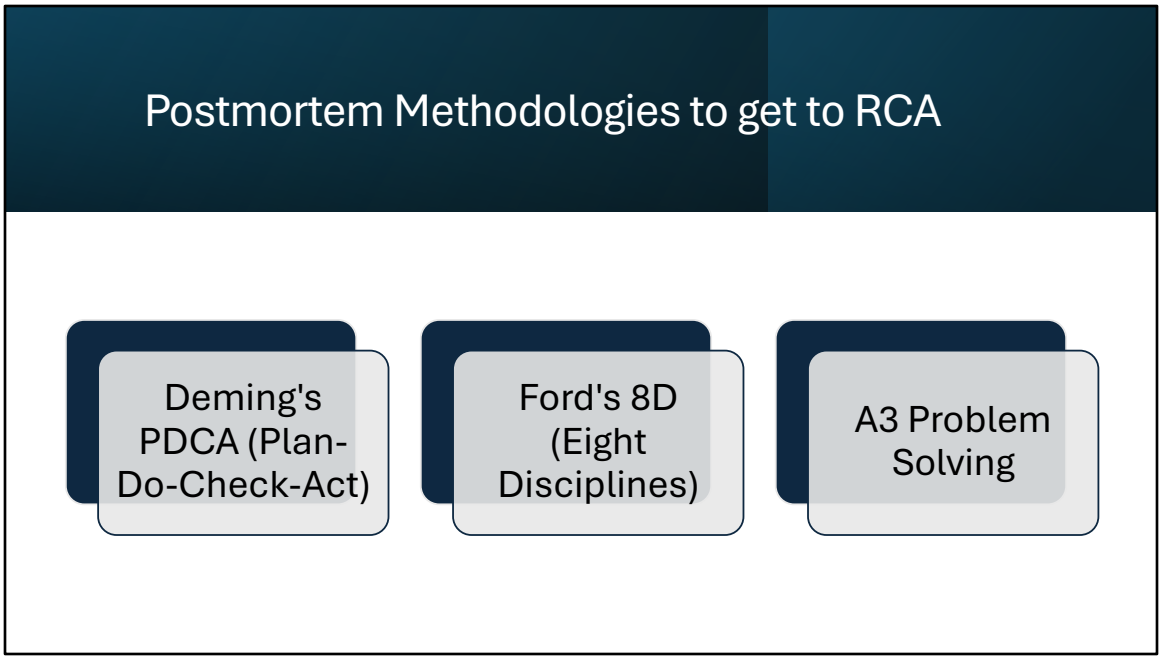
RCA is a structured method used to analyze serious adverse events.
initially developed to analyze industrial accidents,
RCA is now widely deployed as an error analysis tool.
RCA assumes that it is much more effective to systematically prevent
and solve for underlying issues rather than just treating ad hoc symptoms and putting out fires.
Software development is very complex.
The process involved in delivering critical services are not efficient.
we need to be able to understand where the process gaps exist,
identify the root causes, and develop solutions to make us efficient and not repeat errors or failures.
Therefore, it is important to perform an RCA when events happen.

RCA is a subset of postmortem

Basic Steps for RCA



Define problem – define the scope of the incident
Identify root cause – identify all of the potential causes
Determine root cause – find root cause by validating each one
Finding the solution – solution for each root cause
Implement corrective and preventive actions
Measure and assess effectiveness of each solution making sure each root cause is addressed
Yes there can be more than one root cause



PDCA is a four-step iterative process for continuous improvement, introduced by W. Edwards Deming
8D is a problem-solving methodology developed by Ford Motor Company, primarily used in the automotive industry. It involves eight steps
A3 is a structured problem-solving approach that uses a single sheet of A3-sized paper to document the process.

Each of these methodologies has a step that involves RCA

Each methodology has its strengths and is suited to different types of problems and organizational cultures.
PDCA is great for ongoing improvement,
8D is ideal for complex, team-based problem-solving,
A3 is excellent for visualizing and communicating the problem-solving process.

Methodologies to conduct RCA

5 Whys

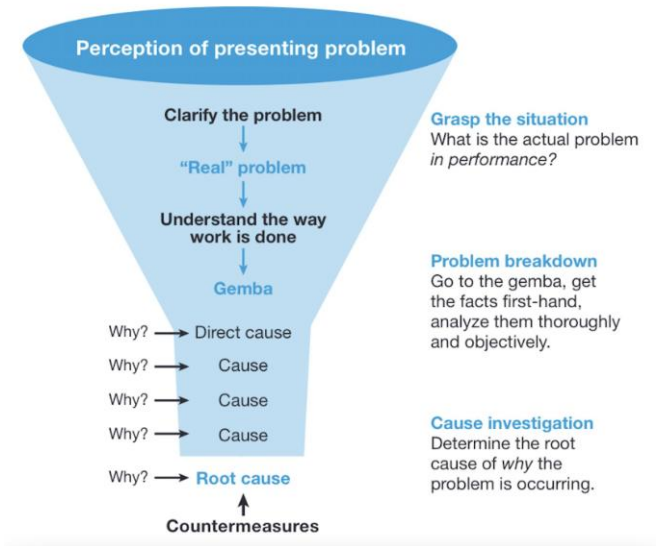
Fishbone diagrams

Visual Map Analysis

Enhanced RCA

- Use one of the following
  - 5 Whys
  - Fishbone diagrams
  - Visual Map Analysis
  - Enhanced RCA

5 Whys

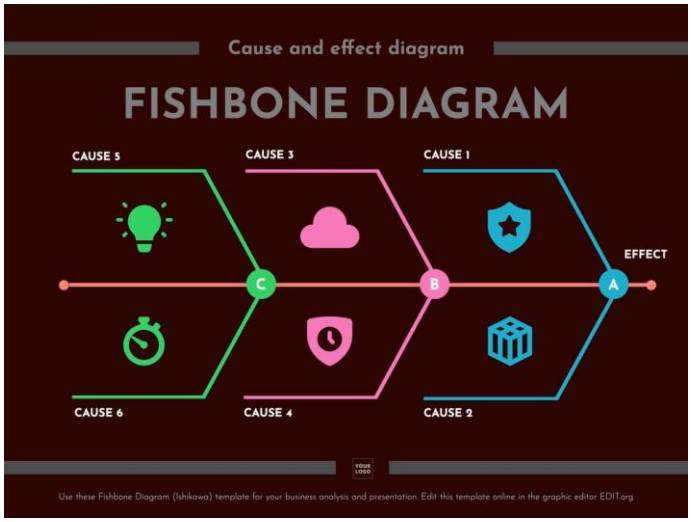


5 Whys is a way to logically dive into sequence of events and identify the root cause.
The problem requires that the team begin with why a problem occurred.
The answer should lead to the next why. The why question is then repeated
repeat the why five times or more until the root cause is identified.
The resolution determines the last why, where an actionable process gap or change has been identified.
The subsequent why question must always be based on the previous answer and should not diverge to a different why question or different topic.
The last why must be back with data and evidence.
There is no what or how and the five whys.

straightforward and easy to use, making it ideal for simple problems.
However, it may not be sufficient for complex issues with multiple root causes.

If you can take away something from this meeting ask 5 Whys when you have an incident
If you at least get to 3 or 4 your team will be much better off

Fishbone

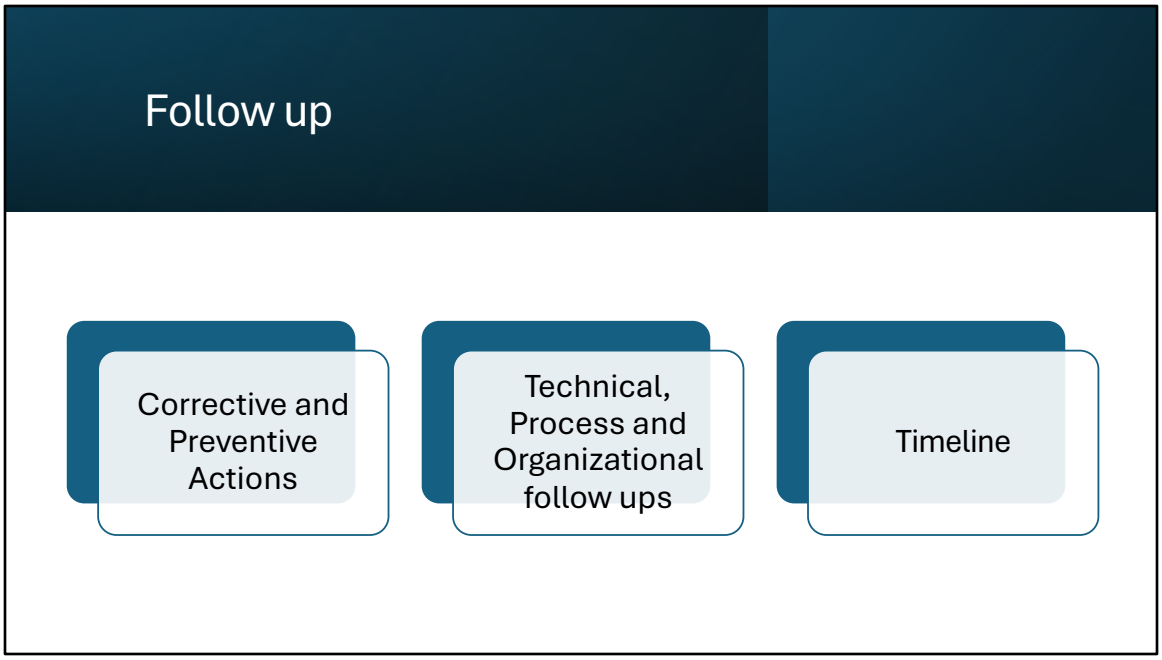


Fishbone Diagram, also known as the Ishikawa or Cause-and-Effect Diagram, is a visual tool used to systematically identify and present possible causes of a specific problem.
The diagram resembles a fish skeleton, with the problem at the "head" and the causes extending as "bones"
Major categories often include methods, machines, materials, measurements, people, and environment

provides a structured and visual approach, making it easier to brainstorm and categorize potential causes.
It's particularly useful for complex problems with multiple contributing factors.

The goal is not to cargo cult a process but to actually take action post-incident

Transforming outages into growth



Understand difference between corrective and preventive actions

Understand the corrective and preventive actions based on types of root causes – technical, process and organizational

Explain corrective and preventive actions in incident and problem management framework often referred to as Repair Items

Corrective and Preventive Actions

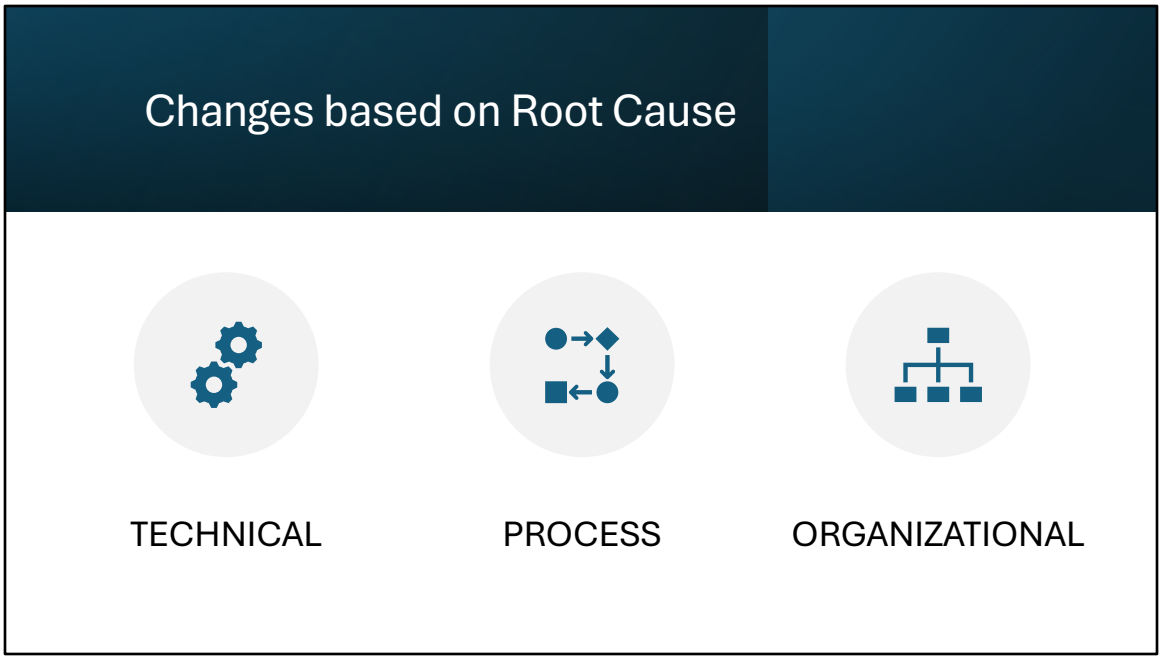
Corrective Actions

- Must address root cause
- Short-term

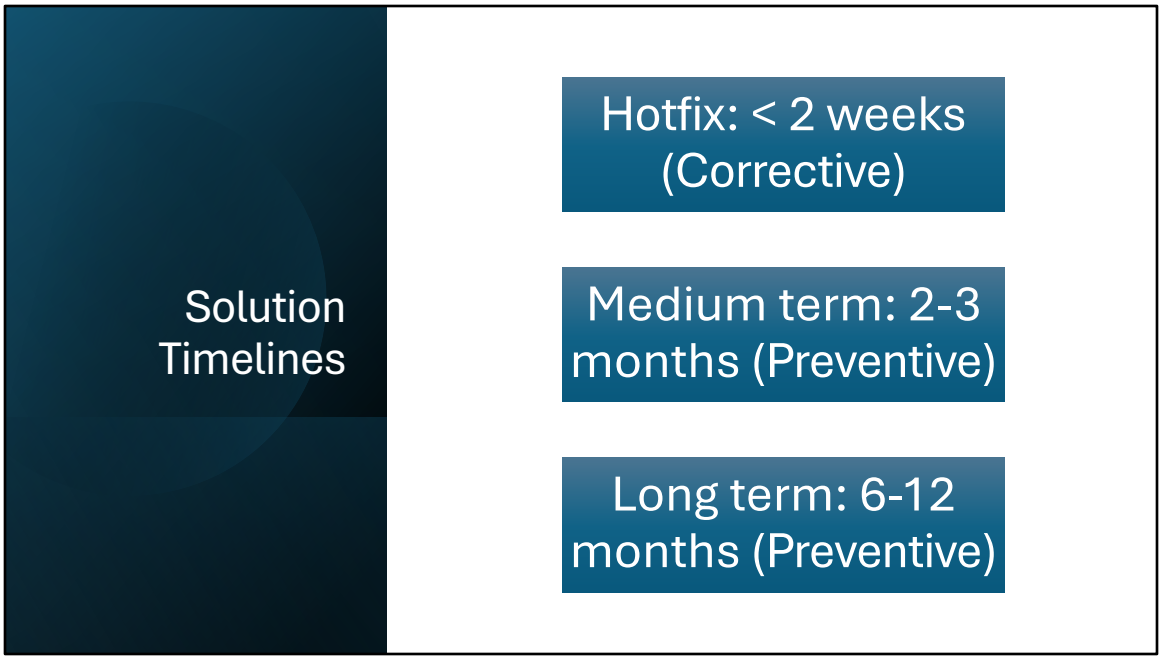
Preventive Actions

- Address future repeat occurrence
- Change Systems, processes and designs
- Medium and long-term

Corrective actions or actions taken to address a problem after the root causes identified, corrective actions must address the root cause.
These are short term actions.
preventive actions are actions taken to address future repeat occurrence of a problem.
preventive actions are needed when systems, processes and designs require changes to address systematic problems proactively
typically these are medium and long term actions.



The most common focus is on the technical however, for systemic issues,
teams must identify processes and orgnizational issues that cause the incidents in the first place.
technical, with a technical cause, you should ask, what is the technical cause of the event?
In most cases it is either equipment or system failure.
Process you should ask what process gaps led to the failure.
Organizational you should ask what management system should have prevented the incident.member,
there are no human to blame here.



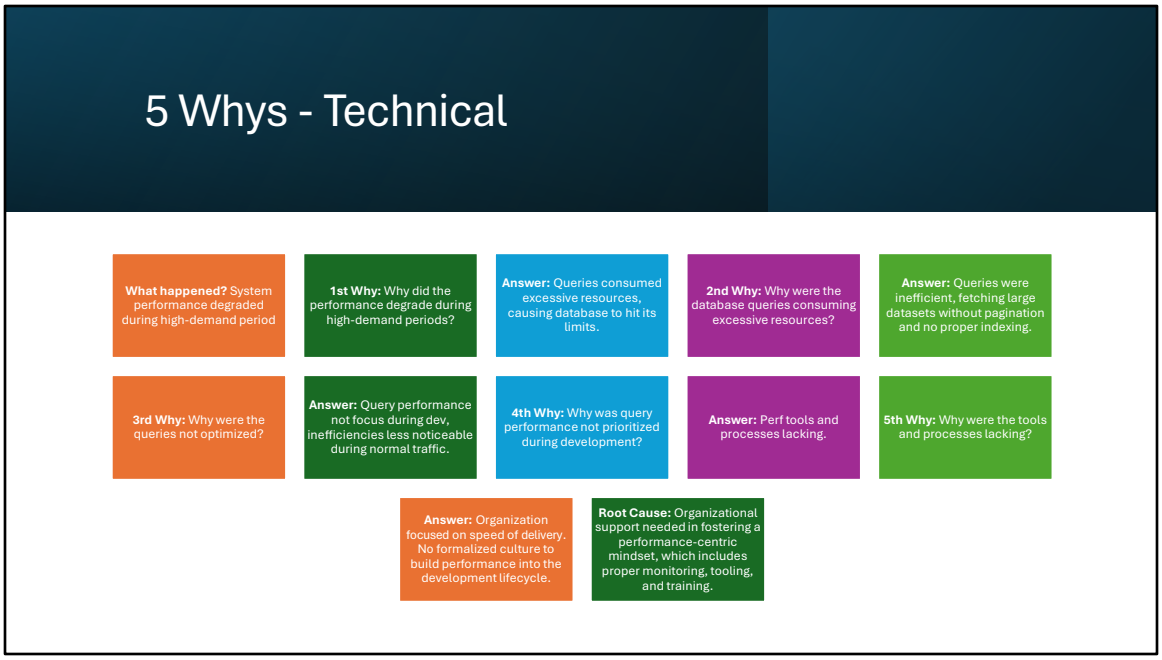
Solution timleines

- Require permanent fixes to address problems to minimize the number of repeat incidents
- Focuses on assessing and analyzing the errors

Timeline

- Corrective: 10 days
- Preventive: 90 days (medium), 365 days (long)

Applying the 5 Whys



What happened? System performance degraded during high-demand period

1st Why: Why did the performance degrade during high-demand periods?

Answer: The database queries consumed excessive resources, causing the Azure SQL Database to hit its resource limits.

2nd Why: Why were the database queries consuming excessive resources?

Answer: The queries were inefficient, fetching large datasets without pagination and lacking proper indexing.

3rd Why: Why were the queries not optimized?

Answer: Query performance was not a focus during development, as the inefficiencies were less noticeable during normal traffic.

4th Why: Why was query performance not prioritized during development?

Answer: Sufficient time and resources not allocated for the development teams to learn and implement tools and processes, to identify and address potential database bottlenecks proactively.

5th Why: Why were the tools, processes, or awareness lacking?

Answer: Organization was focised on speed of deliver for features. There was no formalized culture of continuous improvement or dedicated effort to build performance into the development lifecycle.

Root Cause Identified: Organizational support in fostering a performance-centric mindset, which includes proper monitoring, tooling, and training.

Contract this iwth Johnny Junior Developer wrote bad code, lets pin the blame on Johnny

Process, Tech and Org outcomes for complaints using 5 Whys/similar process

- Tech team response speed was slow
  - •No dedicated staff at time out outage, implement on-call rotation
- Trips staff notified by customer who learned from social media
  - •Fire alerts when errors happen
- Tech team digs through logs from multiple sources
  - •Implement single location with health info and links to logs
- Unrealistic expectations on uptime
  - •Set expectations based on cost-benefit analysis
- Comms were infrequent and not meaningful
  - •Assign manager on-call to handle comms, provide periodic updates with either timeline for next comm or ETA for fix

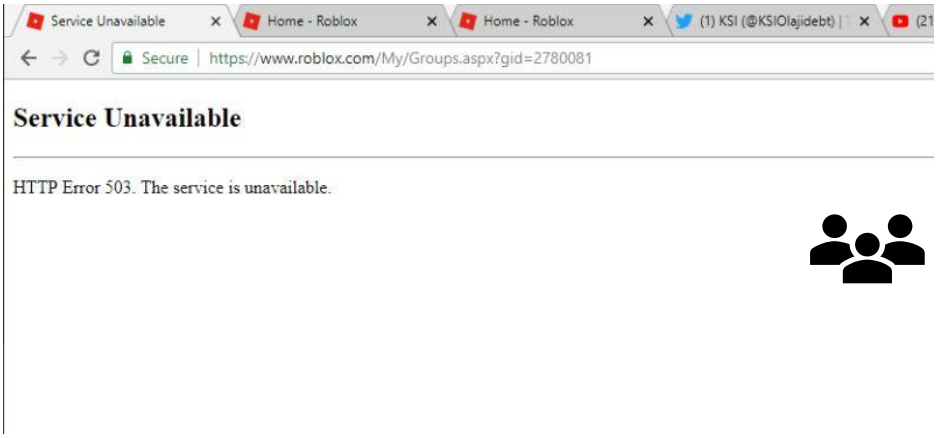
Another outage

Customer flash sale ~8am

Customer Carrie

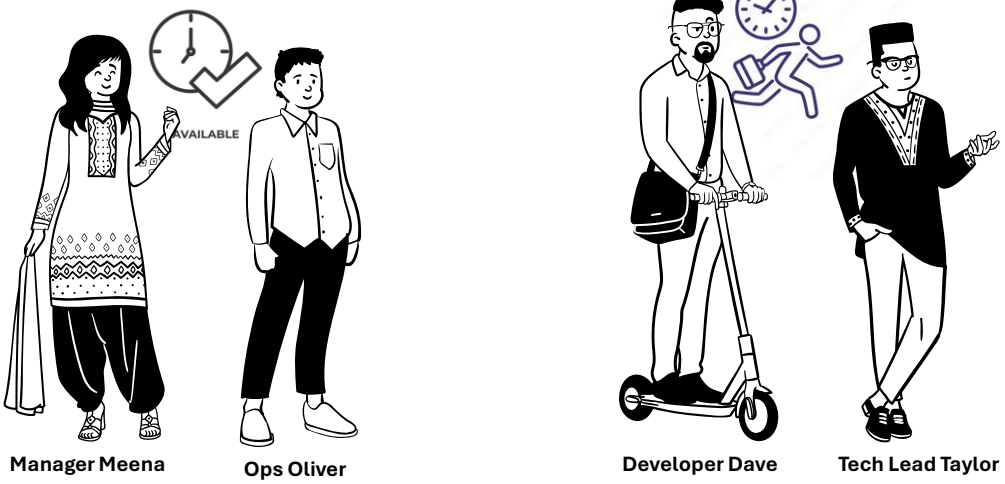
On the Friday morning before a major weekend, when a lot of Trips' staff were out or just taking it easy and logging in late,
one of Trips' biggest customers customer carrie has a flash one day sale
Customer Carrie's campaign lands at 8am in user's mailbox
As a results users login to Trips' website en masse and start hammering the system

Users started seeing intermittent "service unavailable" error ~8:30am

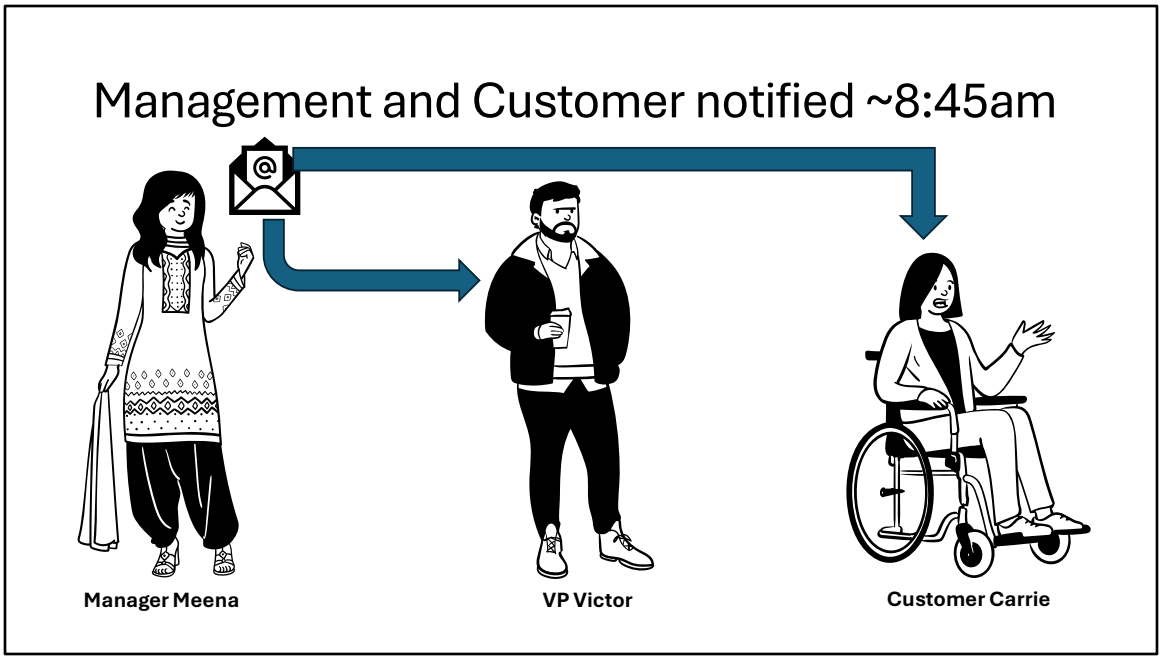


Scaling up seemed to relieve the issue, at least temporarily.
But soon users started seeing intermittent 503 server errors.
This seemed to last for a few minutes, disappear and then rinse and repeat.

Automated notifications ~8:35am

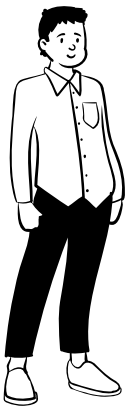


At 8:35am automated email were fired to tech team members
Tech Lead Taylor and Developer Dave were OOF for the day
Manager Meena and Ops Oliver were on-call for the day



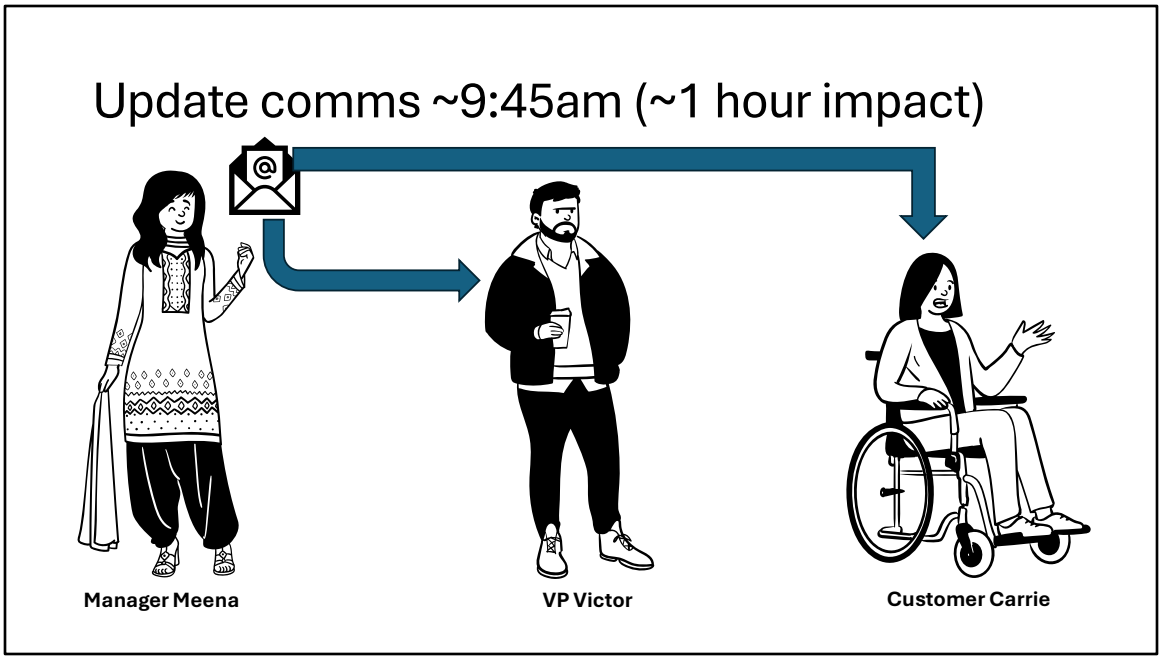
Manager Meena composes two preliminary email
One for VP Victor (internal)
One for Customer Carrie (external)
Basically informs them we've noticed some errors in the customer facing portion of our website
Engineering is looking into it
Next update will be in an hour
Customer Carrie did not even know about the issue and had to check to verify

Ops sees pattern on dashboard ~9:20am

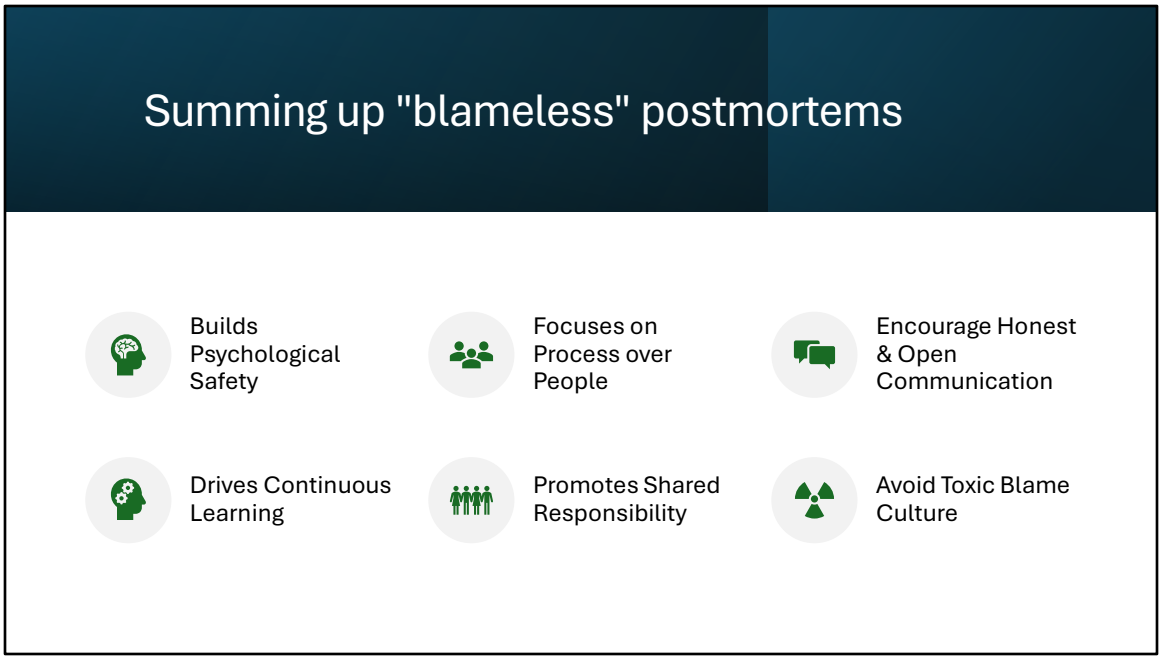


Ops Oliver

Meanwhile Ops Oliver notices on the dashboard
that the scaling is set to 90% CPU usage and everytime it hits 90% there seems to be an outage
The issue seems to be related to not having enough nodes when the incident occurs.
Ops Oliver reduces the threshold to 70%, the issue stops occuring
For good measure Oliver throws in an extra node
Oliver reports back the details to Manager Meena
and shifts to monitoring mode



Manager Meena composes two updates email
One for VP Victor (internal) with all the details of the outage
One for Customer Carrie (external) with reports of the mitigation and promising an RCA in 2 weeks
Customer Carrie while not pleased about the issue, is super happy with the proactive comms
Manager Meena sets a meeting on Monday to start digging into the incident
Tech Lead Taylor on vacation is blissfully unaware of this incident
Monday is another week, the beach beckons



Encourages Honest and Open Communication

- **Description:** Team members feel safe admitting mistakes and sharing insights without fear of punishment or retaliation.
- **Impact:** Ensures that all relevant information is surfaced, enabling a thorough understanding of the incident.

2. Focuses on Systemic Improvement

- **Description:** Shifts the focus from blaming individuals to identifying weaknesses in processes, systems, and tools.
- **Impact:** Encourages teams to see failures as opportunities to strengthen the overall system.

3. Builds Psychological Safety

- **Description:** Creates an environment where team members feel valued and supported, even when things go wrong.
- **Impact:** Enhances collaboration, innovation, and willingness to take calculated risks.

4. Drives Continuous Learning

- **Description:** Blameless postmortems turn incidents into learning opportunities by focusing on what went wrong and why.
- **Impact:** Teams develop a deeper understanding of their systems and processes.

6. Promotes Shared Responsibility
- **Description:** Blameless postmortems emphasize collective ownership of problems rather than finger-pointing.
- **Impact:** Encourages cross-functional collaboration and accountability.

7. Avoids Toxic Blame Culture
- **Description:** Blaming individuals creates a fear-driven environment that discourages transparency and innovation.
- **Impact:** Fear of repercussions leads to hiding mistakes or incomplete incident reporting.

Aligns with DevOps Principles
- **Description:** DevOps emphasizes collaboration, automation, and continuous improvement. Blameless postmortems align with these principles by focusing on iterative learning.
- **Impact:** Facilitates faster feedback loops and more reliable deployments.

Santosh Hari

- Customer Engineer at Microsoft
- Extensive consulting and start-up experience
- Ex-Azure MVP (2016-2020)
- Ex-organizer
  - Orlando .NET User Group (onetug.net)
  - Orlando Codecamp (orlandocodecamp.com)

@santoshhari

@desinole

@santoshhari.dev

End