What would catastrophic failure look like in your organization?

Try and picture this.
Tightly coupled

Loosely coupled

Linear

Complex

Systems that are tightly coupled and complex are less resilient to catastrophe
92% of catastrophic failures are the result of incorrect error handling.
Learning from Failure: Why a Total Site Outage Can be a Good Thing

Alex Elman
Site Reliability Engineer
@_pkill
I help people get jobs.
RAID: Redundant Array of Inexpensive Datacenters
Anycast DNS

US West Response pool failover:
US West » US Central + US East » Global
Failing out a datacenter

Datacenter A
Datacenter B
Datacenter C
Application topology
Hiding broken parts of a service from the user is an example of the **Graceful Degradation Pattern**
Failure Myth #1: It is not worth planning for a catastrophic failure that’s never going to happen
Normalcy Bias

A type of cognitive bias that leaves planners and first responders ill-equipped to deal with or respond to a catastrophic disaster because its occurrence is unencountered or inconceivable.
CATASTROPHE
Indeed is currently unavailable

We should be back online in a few minutes. Thanks for your patience.
Alert: region down
Alert: 4 regions down
Catastrophe
The incident lifecycle

1. Detection
2. Diagnosis
3. Mitigation
4. Recovery
5. Cleanup
6. Retro
7. Prevention
Swiss cheese accident model
Failure Myth #2: A single failure can cause a catastrophe
Diagnosis

```json
OUTAGE: {
    date: "2016-01-20T17:46:07.890-0600",
    description: "Load data artifact",
    errorMessage: "Last load of data artifact failed",
    id: "dataArtifact",
    lastKnownGoodTimestamp: 1453332285462,
    status: "OUTAGE",
    thrown: {
        exception: "RuntimeException",
        message: "Last load of data artifact failed",
        stack: [
            "com.indeed.healthcheck.JasxDependencyManager$15.ping(JasxDependencyManager.java:908)",
            "com.indeed.status.core.PingableDependency.call(PingableDependency.java:59)",
            "com.indeed.status.core.PingableDependency.call(PingableDependency.java:15)",
            "java.util.concurrent.FutureTask.run(FutureTask.java:262)",
            "java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)",
            "java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)",
            "java.lang.Thread.run(Thread.java:745)"
        ]
    }
    timestamp: 1453333567890,
    urgency: "Required: Failure of this dependency would result in complete system outage"
}
```
RAD: Resilient Artifact Distribution

- Use bittorrent protocol
  - Faster
  - Reduced network burden for servers
  - Horizontally scalable
  - Encrypted
- Resilient to network issues
  - Peers in multiple regions/DCs
- Self-service platform
  - Devs can declare data in code
  - No SRE toil needed
Diagnosis

Artifact Generation 1

- JobWebapp
  - artifact.1
- JobWebapp
  - artifact.1
- JobWebapp
  - artifact.1
- JobWebapp
  - artifact.1
- JobWebapp
  - artifact.1
- JobWebapp
  - artifact.1
Load artifact generation 2
Diagnosis

unavailable

artifact.2

JobWebapp

artifact.2  artifact.1

JobWebapp

artifact.2  artifact.1

unavailable

artifact.2

unavailable

artifact.2

JobWebapp

artifact.2  artifact.1

unavailable

artifact.2
Diagnosis

Outage: 0% availability

unavailable

unavailable

unavailable

unavailable

unavailable

unavailable

unavailable

unavailable

unavailable

unavailable
1. Republished artifact to last known good generation
2. Performed a rolling restart of JobWebapp
3. Turned off healthchecking in the load balancer
1. Disabled artifact builder
2. Waited for new artifact to replicate
3. Verified all instances of the webapp were restarted
4. Verified recovery with telemetry
5. Verified healthchecks
This site can't be reached

indeed.com's server IP address could not be found.

Try:
- Checking the connection
- Checking the proxy, firewall, and DNS configuration

ERR_NAME_NOT_RESOLVED
Something is still wrong, go back to diagnosis
The incident lifecycle

Detection → Diagnosis
Mitigation → Recovery
Cleanup → Retro
Prevention
$ host -t A indeed.com
indeed.com has no A record
1. Disabled artifact builder
2. Waited for new artifact to replicate
3. Verified recovery with telemetry
4. Verified healthchecks
5. While waiting for DNS TTL expiration, validated hypothesis
1. Harvested logs and artifacts for investigation
2. Re-enabled healthchecking in load balancer
3. Restored log verbosity levels
4. Restored artifact building
The incident lifecycle

Detection  Mitigation  Cleanup  Prevention

Diagnosis  Recovery  Retro
The incident lifecycle

Detection  →  Diagnosis  →  Mitigation  →  Recovery  →  Cleanup  →  Prevention  →  Retro
Canary Artifact Deployment
1. Attempt to claim a canary lock
2. Load the artifact
3. If successful, “bless” the run
4. After blessing, other servers load the artifact
5. If unsuccessful, log event and try again
Canary Artifact Deployment is an example of the Circuit Breaker Pattern
Failure

Incorporate failure into your system’s design and design for both known and unknown failures.
**Resilience** is a property that describes a system’s ability to adapt to a previously unknown failure while **robustness** is a system’s ability to recover from a known failure.
Failure Myth #3: Failure can be prevented
Failure is a routine part of running distributed systems
Systems that are tightly-coupled and complex are less resilient to catastrophe.
Global aviation is an example of a complex and tightly coupled system
Failure Myth #4: Adding resilience improves reliability
Improving resilience can reduce reliability.
Incident Response
Situational awareness in decision making
Our cognitive biases are useful adaptations but they often lead us astray during incident response.

You don’t have to eliminate them but be aware of them.
Availability heuristic
Relying only on the ideas that come to mind when making decisions in uncertain situations.
Focusing effect
The tendency to place too much importance on one aspect of an event.
Illusory correlation

Inaccurately perceiving a relationship between two unrelated events.
Confirmation bias

The tendency to search for, interpret, focus on or discard evidence in a way that confirms one's preconceptions.
The incident lifecycle revisited

Detection
Diagnosis
Mitigation
Prevention
Cleanup
Retro
Recovery

New symptom emerges
Responder joined later
Dev helped with identifying DNS issue in Slack
indeedhi.re/2wKa2Mm
The Retrospective Process
The Retrospective Process
Learning from incidents

Retrospective start

- Urgent remediations addressed
- Debriefings
- Synthesis & Analysis
- Retro report compiled
- Retro report released
- Remediation meetings
The Retrospective Process
Learning from incidents

Testimony is most accurate within two weeks of return to normalization.
Debriefing
Debrief attendees

+ Debrief facilitator
+ Debrief facilitator trainee
+ Scribe
+ Incident owner
+ Incident participants
+ Retrospective owner
+ Subject matter experts
Qualities of debrief facilitators

+ **Impartial**: Not involved in the incident
+ **Curious**: Asks questions
+ **Attentive**: Listens
+ **Respectful**: Improves psychological safety
+ **Thorough**: Captures all relevant testimony
+ **Patient**: Mediates heated debate
+ **Uses shared language**: Sufficiently technical
Debrief agenda

1. Facilitator reviews the timeline
2. Facilitator interviews attendees
3. Call for clarifying questions
What questions should a facilitator ask?
What was happening just before the incident?
During the incident?
Was there a call for assistance?
How was it known who to contact?
How could this incident have been worse?
How did we arrive at the decision to turn off the healthchecking in the load balancer?
Debriefing tips

+ Start debriefs as soon as possible
+ Before the debrief
  + Send out questions to participants
  + Assess the comfort level of participants
  + Commit someone to scribe or record
+ Conduct 1:1 debriefs if necessary
The Retrospective Process
Learning from incidents

Interviews, narratives, contributing factors, latent threats, impact, remediation items

- Address urgent remediations
- Debriefings
- Synthesis & Analysis
- Retro report compiled
- Retro report released
- Remediation meetings
Avoid counterfactuals

+ “...made a mistake by…”
+ “The developer carelessly…”
+ “... suboptimal decision-making…”
+ “... should have been obvious…”
+ “Could have prevented the outage…”
+ “... failed to verify the change…”
Root cause analysis is a fairy tale
Root cause is also an **imprecise** concept.
1. Initiating cause
2. Most basic cause
3. Earliest cause
4. Deepest cause

WIKIPEDIA’S DEFINITION OF ROOT CAUSE

So many choices...
1. **Initiating:** Non-critical healthcheck dependency commit?

2. **Most basic:** Filesystem exhaustion on build server?

3. **Earliest:** The Big Bang??

4. **Deepest:** The Human Condition???
Root cause analysis is **too narrow** in scope to maximize learning.
Root cause analysis is too narrow in scope to maximize learning.

It leaves important contributions unexplored.
Root cause analysis is not blame-aware.
The Five Whys is also problematic
Is the root cause hiding here somewhere?
Fixating on root cause is an easy trap to fall into.
Causal analysis and diagnosis are supremely important activities.
What should we do instead?

Locate **contributing factors**
Contributing factors

- Artifact publishing script didn’t handle a certain exception
- Builder used non-atomic filesystem writes
- Filesystem filled up to 100%
- Non-critical healthcheck dependency marked as REQUIRED
- No fail-open pool in the DNS traffic director
- Corrupt data artifact loaded into webapp without verification
The Retrospective Process
Learning from incidents

- Address urgent remediations
- Debriefings
- Synthesis & Analysis
- Retro report compiled
- Retro report released
- Remediation meetings

Write report and assemble deliverables
1. Contributing factors
2. Remaining threats
3. Remediation items
4. Command line history
5. Chat transcripts
6. Graphs
7. Retrospective report
The Retrospective Process
Learning from incidents

Promote this material far and wide in your organization. Add this to your incident library.
The Retrospective Process
Learning from incidents

These happen on the team level. This is where remediation owners are determined.
Execution is team dependent

Dive deep retrospective report

Assign owners for remediation items

Discuss finer points of the contributing factors

Can continue in perpetuity
We don’t deeply know our systems.
System as imagined

urgency: "Weak: Failure of this dependency would result in minor functionality loss"

System as found

urgency: "Required: Failure of this dependency would result in complete system outage"
Failure

The best opportunity to gain an understanding about how our systems behave is through failure.
Chaos testing

Test in ALL environments with the goal of validating your hypothesis.

Discovering things you didn’t know about your systems is a consequence.
Failure Myth #5: Safety can be measured by the number of accidents that occur
WHAT CAN BE MEASURED?

+ Number of threats identified and mitigated
+ Number of tests running (including prod)
+ How readily information travels through an organization
+ How reliability work is prioritized compared to feature work
+ How experience with failure influences future design decisions
+ Psychological safety
1. Embrace failure as part of your systems
2. Evolve into a learning organization
3. Move the boundary of your systems to include people who interact with them
4. Humans are imperfect responders. Be aware of your cognitive biases.
5. Root cause analysis hinders learning and is not blame-aware. Locate contributing factors.
Revisit what catastrophic failure looks like for you. Why isn’t this happening in your organization right now?

Do you know what’s going right and why?
Thank you