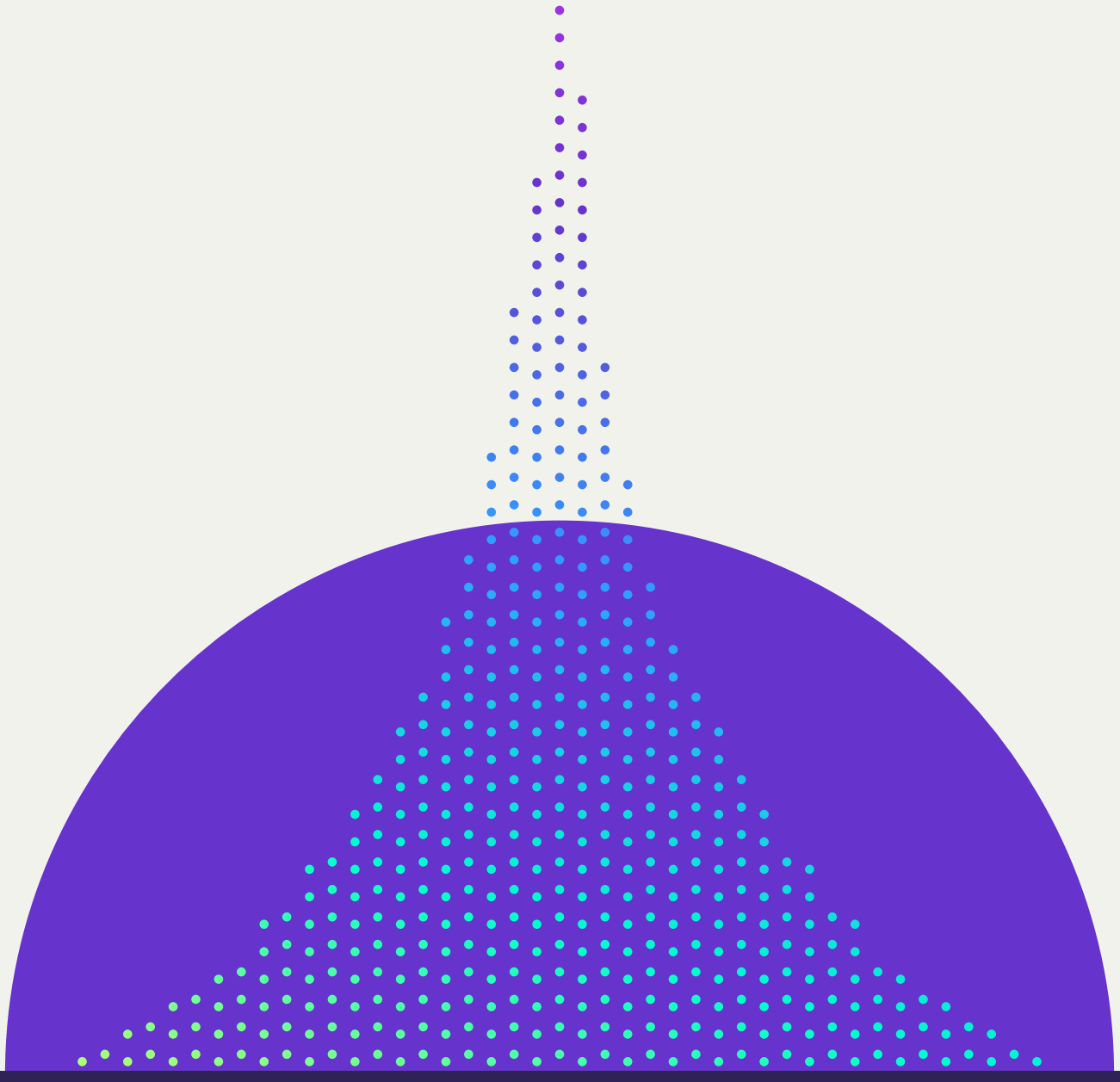
Gremlin

Preparing for Large Traffic Events

A CHECKLIST FOR LOGISTICS, RESOURCES AND PROCESSES



Summary

When we know there will be a higher-than-usual amount of traffic to our web application, many of us get nervous, and with good reason. Large failure events on these days have cost companies **large amounts of money**, like $11 million for **Costco** in 2019 and $700 thousand for **J.Crew** in 2018.

It's not enough to set up our systems and hope everything is going to work, even when we have put significant time and effort and skill into a good design. We can't know how our system will react to events like these without performing some preparation work and testing.

This checklist will help companies prepare for Black Friday, Cyber Monday, and similar high-traffic events to become one of the success stories, like one of Gremlin's customers since 2018, **Target**, whose site crashed on **Cyber Monday in 2015** along with many other retailers. Target's site has become more reliable each year while using techniques like those in this checklist, including the addition of Reliability Management and Chaos Engineering by many of their teams since 2017.

Pre-Checklist:

DESIGN/ARCHITECT FOR FAILURE

Is your system designed for reliability? Reliable systems include things like load balancing, autoscaling, and failovers. Have you found and removed single points of failure? Thought of and implemented all the mitigation schemes you can think of (or afford)?

Do what you can early on and it will prevent much pain later. Most companies plan to do a soft code freeze (bug fixes only, no new features) about 8 weeks before a planned big event and a hard code freeze about 2-3 weeks before.

Companies that are unsure about their systems or have less experience and resources may freeze sooner. Companies with more experience may freeze later. In any case, any preparation and testing must be done before the hard freeze. When the hard code freeze before a big sales day arrives (or other large traffic events like online testing days or the Super Bowl or even Valentine's Day), it is too late for a redesign. You must do what you can today, so let's jump in.

Logistics

In this part of the checklist, take a close look at the practical details outside of cyberspace to see if you are prepared. Start by creating a document that includes a minimum of these things. Then, get all assets set up and ready before the ramp up starts.

NAME

Define the event. Is it Black Friday and Cyber Monday? Are you selling pizza during the Super Bowl? Is it time to deliver online tests to students across an entire state?

TIME

Set a beginning and end date and time that you expect the high traffic event to occur, wide enough to encompass the entire window. Note: this should start 48-72 hours before the event for ramp up and end 48-72 hours after the event to handle trickle down.

PEOPLE

Identify which team(s) will be on call and make sure everyone on those teams is ready. Communicate to everyone else to be alert to special needs during that window, even if they are not on call.

- [ ] Identify the incident manager on-call (IMOC)
- [ ] Define the on-call rotation
- [ ] Notify everyone on the on-call team(s)

COMMS

Set up a war room for the duration of the event. This is where all activity will be coordinated and where all reports will be sent. The war room includes a physical space, and not everyone involved has to be in the actual room all the time. This is just the cross-team leadership and for specific team leaders to meet with one another as needed to coordinate efforts. There should also be virtual war room assets, such as listed below.

- [ ] The physical space can be a conference room which will require plenty of supplies, stable and fast internet access, water, snacks, and an occasional pizza order.
- [ ] Create a dedicated instant messaging channel using Slack, Hangouts, Skype, or whatever your company currently uses. Test and make sure all involved are invited and can access the channel.
- [ ] Create a dedicated video conference using Zoom, BlueJeans, Webex, or whatever your company currently uses. Test and make sure everyone involved can access.

Resources

IDENTIFY ALL INFRASTRUCTURE RESOURCES INVOLVED

- [ ] Where is the data center (DC)? The region? The availability zone (AZ)? The failover/backup/etc.? Know and document every part of the system. A tool like Gremlin can help identify critical services, dependencies, and the underlying infrastructure.
- [ ] App groups?
- [ ] Any other resources (image store, database, etc.?)

CREATE RELEVANT DASHBOARDS AND ALERTS

Do this for each of the most meaningful statistics you monitor. Some example metrics include:

- [ ] Kubernetes pod or Docker container status or virtual machine (VM) health (up, running, etc.)
- [ ] Pods/containers per node/host
- [ ] Pods/containers ready vs needed vs unavailable
- [ ] Service/microservice instances status, endpoint availability, etc.
- [ ] API status (especially Third Parties you're relying on)
- [ ] Usage (total running sessions and total unique sessions, Real User Monitoring tools are very helpful)
- [ ] Request volume (client and server) and success rate, maybe requests by source
- [ ] Response time for requests
- [ ] CPU/memory/storage utilization
- [ ] Database/storage responsiveness
- [ ] Dependency failures
- [ ] Malware/DDoS assessment
- [ ] In the E-Commerce space, orders per second/minute/hour

CREATE/RE-ITERATE APPLICATION AND INFRASTRUCTUREARCHITECTURE

CAPACITY PLANNING

- [ ] Pre-warm the front door (scale up Edge, API Gateway, Webservers, etc.)
- [ ] Check/exercise scaling policies (if you don't test them, you don't actually know if they will work as designed!)
- [ ] Identify how traffic is handled at every app layer: Kubernetes, integrations, clusters, technologies and dependencies, application services, load balancing and failover systems (Application Performance Monitoring tools & Distributed Tracing are super helpful here)

Processes:

HAVE A MASTER RUNBOOK

for each team set up with its location communicated team-wide. Don't put it in the war room, because that location is intended for coordinating across multiple teams, if needed; put it wherever the team usually works, including the possibility of it being in a clearly-communicated online location for remote teams.

VERIFY YOUR DISASTER RECOVERY (DR) PLAN

- Validate recovery mechanisms (such as DC failovers, and app mitigation paths like circuit breakers)
- Check dependencies, implement paths to mitigate single points of failure, or escalation/oncall path
- Test the escalation path

TEST IN PRODUCTION

Ultimately, there's no way to know how your systems will perform when it matters most unless you're testing where your customers live: in a real-world environment.

This testing can be done safely and securely with a **Reliability Management Platform** such as Gremlin. Based on reliability best practices and Chaos Engineering principles, Reliability Management helps teams **orchestrate pre-defined tests** to validate against common reliability risks and proactively understand their overall reliability posture.

Companies like **Backcountry** and Target test their entire system using Gremlin throughout the year, using the results of testing to enhance system reliability and prioritize improvements. They also test their systems in specific, prioritized ways in advance of known events like those described here, looking at specific functionality like add-to-cart to see how it will hold up to increased load.

Gremlin

Gremlin's Reliability Management Platform enables high-velocity engineering teams to standardize and automate reliability across their organizations without slowing down software delivery. Gremlin's Reliability Score sets the standard for reliability so there's no guesswork, and an automated suite of Reliability Management tools makes it easy to integrate reliability throughout the software lifecycle so there's no slowdown.