SHARE TECH 19/10/21

The Delivery Hero Reliability Manifesto

By Christian Hardenberg

Our Reliability Manifesto is a succinct collection of rules, guidelines, and best practices that reflect our current thinking on what it takes to build a reliable system.

At Delivery Hero, we solve complex challenges at scale – from how to get an ice cream

DeliveryHero Tech

We currently process close to 10 million orders every day. Three things make this especially difficult:

1. Our peak time is Sunday night, a time when our tech team likes to sit at the dinner table rather than in front of a Macbook.
2. As food gets cold quickly, we have only minutes to recover from problems.
3. If things don't work as expected, we are responsible for millions of restaurants losing business and a lot of hungry customers.

Adding to the challenge, we have seen exponential growth in the last few years, with some countries growing >1000% year-on-year, forcing us to re-architect large parts of our systems towards no-sql data layer, non-synchronous communication, and a multitude of new services.

We did all this in a rather decentralized way, without a common infrastructure team. Instead, we relied on hundreds of technical leaders across squads, domains and tribes to do the right thing.

But what is the right thing to do? It reminds me a bit of my early days as C++ programmer at ART+COM (now to be seen on Netflix by the way), when there were many traps to avoid, like dereferencing a null-pointer which would miserably crash your system. Back then I was so happy to be handed a document called "C++ Coders Practices" with 109 very helpful rules.

At Delivery Hero, we sat down with our tribe leadership, aiming to achieve something similar for system reliability. To help our brave tech leaders "do the right thing" we created a seven-page document called the **"Delivery Hero Reliability Manifesto".**

As the document has now gone through a number of revisions and reached a good level

The manifesto is a succinct collection of rules, guidelines, and best practices that reflect our current thinking on what it takes to build a reliable system.

When writing the document, we decided on an assertive style. The goal was to keep things short, clear and easy to implement.

The document alone is only part of the story of how we achieve high reliability. Other important things that we do to get there include:

- Co-creating such documents by involving a lot of smart people.
- Constantly incorporating new learnings, with ongoing debate and regular document revisions.
- Gamifying and incentivising teams, by translating our learnings into a "reliability score", which we publish for each service.
- A weekly operations meeting to share learnings from failures, as well as celebrate successes.
- Keeping track of our technical debt and paying it back eventually.
- Hiring a diverse group of smart, pragmatic, caring engineers, who together can move mountains.

If you are considering joining Delivery Hero as an engineer, this document can give you a glimpse into how we do things.

If this makes you curious to know more and you are interested in joining us on our mission to deliver amazing experiences to millions of customers across the world every day, you can find plenty of great opportunities here.

Christian

DeliveryHero Tech

Meta Rule

M-1 We do not silently bypass the rules in this document, but instead, start a discussion to change the rules.

M-2 We assess the validity of the rules in this document twice a year.

Architecture & Technical Debt

A-1 *We document architecture:* All services and their dependencies must be documented in an architecture chart, following the conventions described in our Architecture Guideline.

A-2 *We care about naming:* All services must have unambiguous and easy to understand names.

A-3 *We start with a monolith:* Following the monolith first pattern allows us to move fast and break into sub-services at a later point when we have a better knowledge of requirements.

A-4 *We embrace microservice principles*. New services must:

- Only communicate via APIs or messages.
- Not share data storage with other services.
- Be part of the product domain (a bounded context) not a technical component.
- Follow the single-responsibility principle and do one thing well.
- Be owned by only one squad.
- Avoid synchronous call-chains deeper than 2 services

A-5 *We embrace the data mesh:* Data producers are responsible for publishing their domain data as a product on our common data platform (Datahub), ensuring they are:

1. Discoverable.
2. Addressable.
3. Trustworthy and truthful.
4. Self-describing semantics and syntax.
5. Inter-operable and governed by global standards.
6. Secure and governed by a global access control.

A-6 *We review architecture:* New services must pass a technical design review meeting with the tribe & vertical tech lead. Tribe tech leads present their overall architecture to peers biannually. Both meetings are open to anyone interested.

A-7 *We know our debt*: We agree on technical debt quarterly and track it.

A-8 *We are cloud native:* Where possible we choose managed services over self-hosted alternatives.

A-9 *We minimize critical things:* We avoid services to be part of the critical path to fulfil an order. We keep critical services lean by moving non-critical scope somewhere else.

A-10 *We document decisions:* We use RFCs and architecture decision records to align on and document important architectural decisions along with their context and consequences.

A-11 *We love well defined APIs:* We document every API open to other teams and publish the docs in our API library.

D-2 *We track deployments*: All deployments must be tracked via deployment events.

D-3 *We are open 24/7:* All services must support zero-downtime deployment. Deployments during peak times and Fridays are safe.

D-4 *We deploy with a safety-net*: All Tier 1 services must support automated canary deployments.

D-5 *Our infrastructure is code*: All infrastructure must be configured in code and in the repository. All infrastructure changes run through a CI/CD pipeline.

D-6 *We automate testing*: We expect production code to have a 60-80% code coverage with unit tests. In addition the main application flows must be covered by integration tests.

D-7 *We never code alone*: For new and complex code we do pair programming. All production code changes go through a code-review.

Resilience

R-1 *We limit risk*: Countries larger than 10% of our business (measured in GMV) must run their critical services on dedicated resources.

R-2 *We don't treat things equally:* Every service is assigned to one of the following tiers:

1. Tier 1: Core services affecting >20% of overall business within the first 30 mins of unavailability. Additionally, all services required to process an order (i.e. checkout, cart & vouchers, payment, order transmission & fulfillment).
2. Tier 2: Other services where a downtime has direct bottom-line impact via loss of

R-3 *Our services degrade gracefully*: Tier 1 and 2 services must not go down if a dependency fails. Instead, they must fall back to default behaviour with degraded mode.

R-4 *We design for failure:* We implement the following protection mechanisms:

1. *Timeout*: Beyond a certain wait interval, a successful result is unlikely or simply too late.
2. *Retry*: Many faults are transient and may self-correct after a short delay – use exponential backoff and jitter to avoid cascading failures.
3. *Circuit Breaker:* When a system is seriously struggling, failing fast is better than making clients wait to avoid running out of critical resources.
4. *Fallback*: Things will still fail – plan what you will do when that happens.
5. *Throttling*: Prevent misbehaving clients bringing your service down.
6. *We prepare for rapid growth with ongoing load-tests:*ts bringing your service down.
7. *Idempotence*: Multiple identical requests made to a service apply only once.
8. *Recoverable:* Your service has a process to replay messages to recover from an outage of a downstream service.
9. *Dead-letter queues:* Erroneous messages do not stop a service processing valid messages and are published to a dead-letter queue for later analysis.

R-5 *We test for failure*: Tier 1 and 2 services must test for dependency failures. Synchronous failures include no response, slow response, error response and unexpected response (e.g. bad JSON, missing mandatory fields, wrong types). Asynchronous communication, in addition, can be duplicated or out of order.

R-6 *We track the golden signals*: Every team needs to have a real time dashboard showing, at a minimum, requests per minute, error rate, server response time and a business metric that is highly correlated with system health.

DeliveryHero Tech

R-8 *We log in a central location:* All logging goes into a single place using shared conventions. We are aware of the cost of logging and make conscious decisions about what needs to be logged.

R-9 *We prepare for rapid growth with ongoing load-tests:*

1. *Scale:* We take the peak requests per minute of the last week and verify that we can ramp-up within 1 min to 3x, and within 30 min to 4x, starting from average load.
2. *Frequency:* At a minimum bi-weekly.
3. *Quality:* We use realistic request patterns including write operations.
4. *Environment:* Wherever possible, we run load tests on production.
5. *Expectation:* A load test is successful if server response times and error rates stay within acceptable levels for the particular service.

R-10 *We agree on error budgets:* The agreed budget for lost orders due to incidents is 0.1%. We allocate 0.05% to our platforms and 0.05% to the global services. We track and report actual numbers vs budget monthly.

R-11 *We cancel noise:* Our public APIs have a maximum error rate of 0.01% (daily average).

R-12 *We speed things up with runbooks:* We document steps to analyze and react to issues in runbooks and link them directly to the alert to minimize time to recovery.

R-13 *We prepare for chaos:* Services have documented all dependencies and defined mitigation strategies for dependent service failures. We regularly test the most important/impactful ones.

R-14 *We don't rely on compute-instances*: Any compute-instance can be killed at any

R-16 *We can recover from disasters:* All data must be backed up and stored in long term cloud storage in encrypted form with a minimum of 30 days retention. Tier 1 and 2 services must prove that they can recover from a disaster within 2h.

Continuous Improvement

C-1 *We track all incidents:* Major and critical incidents are reported and updated using our statuspage. All incidents are tracked via a tracking sheet.

C-2 *We learn from our mistakes*: All major and critical incidents must be documented with post-mortems following a template and published on statuspage. Incidents are reviewed in weekly stability review meetings.

C-3 *We go to WOR:* During Weekly Operational Review, we work together to improve the reliability and security of our systems. We share learnings from incidents, success stories, business and technical updates, and also randomly pick a squad via the wheel of fortune and ask for:

1. A concise introduction of yourself and what your team does.
2. An architecture chart that shows your services and highlights which services are critical.
3. Share any interesting incidents in the last 3 months and what others can learn from it.
4. Show your monitoring dashboard (Requests per Minute, Error Rate, Server Response Time, Business Metrics).
5. Share how you are doing on load tests, and what the peak RPM multiplier you can safely reach.
6. Present the top 3 things that could go wrong and how you plan to mitigate them.

C-4 *We share our code*: Every team has the right to contribute to any code of Delivery

DeliveryHero Tech

C-6 *We foster honest and open communication*: We are transparent and we have discussions based on facts, where the best argument should win.

C-7 *We like diversity of people and ideas:* Every voice matters.

C-8 *We assume good intentions:* We interpret written communication always in the most charitable way.

Security

S-1 *Security is not optional:* We protect our systems and data with the highest diligence and constantly improve our security dashboard rating.

S-2 *We block attackers at the edge:* All customer-facing DNS entries must be protected with Cloudflare. Inter-service communication must be pro.

S-3 *We limit our attack surface*: Direct access to internal tools, back-offices and infrastructure is not possible from the public internet.

S-4 *We control access in one place*: Over time all back-offices will plug into OpsPortal and use DH IAM service for identity and role management.

S-5 *We lock away our secrets*: All secrets are managed via a vault solution (Vault is recommended). We never store secrets on Github.

S-6 *We simulate attacks:* For publicly available services we ask the DH security team to perform a penetration test twice a year (via security@deliverohero.com).

S-7 *We respect privacy:* We handle personal data in accordance with the GDPR

DeliveryHero Tech

2. Minimizing the collection of personal data at the outset to only what is strictly necessary.
3. Limiting the use of personal data to the specific purposes for which it was collected.
4. Avoiding the identifiability of individuals through the use of pseudonymization and anonymization.
5. Ensuring the confidentiality, integrity and availability of personal data by using state-of-the-art technical and organizational measures.

S-8 *We gamify security:* In order to compare teams and incentivise the right behaviour, we calculate and publish a security high score table.

Performance & Cost Efficiency

P-1 *We don't waste time:* All services serve 95% of requests under 150 milliseconds.

P-2 *We make cost visible on squad level:* We use resource tagging on squad and service level for cloud resources, observability tools and data processing. All squads must track their weekly costs.

P-3 *We leverage economies of scale:* We build our systems so that cost per order decreases with increasing order numbers. We expect at least a 10% downward trend Quarter-on-Quarter.

P-4 *We are frugal with logging*: Observability cost should not exceed 15% of our total cloud cost.

Elite Level Rules

[illegible]

DeliveryHero Tech

E-3 *We kill our mutants:* We do mutation testing to validate the quality of our unit tests.

E-4 *We only trust what is tested:* We integrate performance and security testing in your CI/CD pipeline.

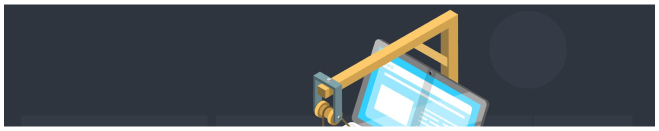
E-5 *Every microsecond counts:* We use gRPC/Protobuf protocol for service to service communication.

E-6 *We trust each other:* We use trunk based development to encourage pair programming, collective code ownership and increase code visibility within the team.

E-7 *We know our metrics:* We track lead time (the time it takes to go from starting development, to the deployment being made, to production), deployment frequency, meantime to restore, and change fail percentage.

As always, we are still hiring, so check out some of our latest openings, or join our Talent Community to stay up to date with what's going on at Delivery Hero and receive customized job alerts!

MOST POPULAR



How we migrated a global and business-critical API with zero down...

DeliveryHero Tech

Welcoming PyBerlin to Delivery Hero

Delivery Hero Architecture Reviews

The Delivery Hero Reliability Manifesto

Preparing for your Technical Interview @ Delivery Hero

DeliveryHero Tech

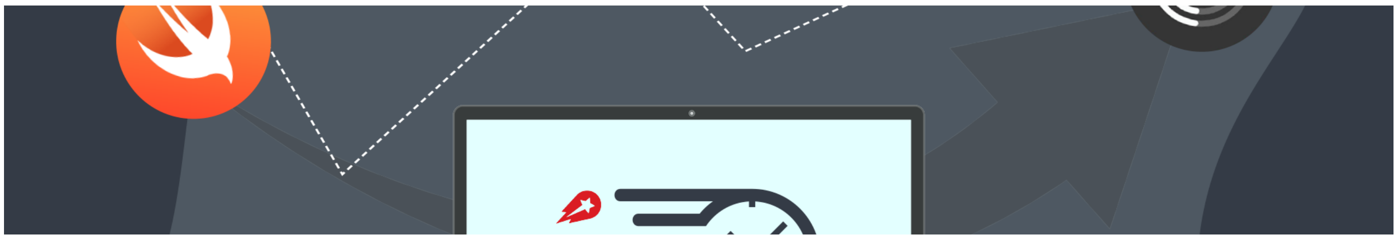
WRITTEN BY:

Christian Hardenberg
CTO at Delivery Hero

LATEST ARTICLES

BACKEND By Yonggeun Kwon

Building event-based architecture for member system



DeliveryHero Tech

Why do the same thing over and over? External dependencies, Swift Package Manager

INFRASTRUCTURE/SYSTEM

By Wesley Costa

Creating an SRE Culture while preventing a 12 million order loss

BACK TO ALL NEWS

DeliveryHero Tech

GET IN TOUCH WITH US

Oranienburger Straße 70
10117 Berlin Germany

DeliveryHero Tech

FOLLOW US ON

Imprint Disclaimer Security About Newsroom Careers Contact