Chapter 7

Operations in a Distributed World

Professor Rieks
Fall 2022
IST 643

Operations Defined

- **Operations**: the work done to keep distributed systems running.
- Must be done that meet or exceed the Service Level Agreements (SLA’s)
- Focuses on
  - Availability
  - Speed / Performance
  - Security
  - Capacity Planning
  - Software / Hardware Upgrades
- Google / others call this the **Site Reliability Engineering (SRE)**

Terms to Know

- **Innovate:** Doing (good) things we haven’t done before.
- **Machine:** A virtual or physical machine.
- **On-call:** Being available as first responder to an outage or alert.
- **Server:** Software that provides a function or API. (Not a piece of hardware.)
- **Service:** A user-visible system or product composed of one or more servers.
- **Soft launch:** Launching a new service without publicly announcing it. This way traffic grows slowly as word-of-mouth spreads, which gives operations some cushion to fix problems or scale the system before too many people have seen it.
- **SRE:** Site Reliability Engineer, the Google term for systems administrators who maintain live services.
- **Stakeholders:** People and organizations that are seen as having an interest in a project’s success.

Operations

- To understand distributed systems operations, one must first understand how it is different from typical enterprise IT. One must also understand the source of tension between operations and developers, and basic techniques for scaling operations.
- **SRE versus Traditional Enterprise IT**
  - SRE is different from an enterprise IT department because SREs tend to be focused on providing a single service or a well-defined set of services. A traditional enterprise IT department tends to have broad responsibility for desktop services, back-office services, and everything in between (“everything with a power plug”).
  - SRE’s customers tend to be the product management of the service while IT customers are the end users themselves.
  - SREs tend to manage services that are constantly changing due to new software releases and additions to capacity. IT tends to run services that are upgraded annually (if that).

Change vs Stability

- There is a tension between the desire for stability and the desire for change. Operations teams tend to favor stability; developers desire change.
- A developer is praised for writing code that makes it into production.
  - Changes that result in a tangible difference to the service are rewarded above any other accomplishment.
  - Therefore, **developers want new releases** pushed into production often.
- Operations, in contrast, is rewarded for achieving compliance with SLAs, most of which relate to uptime.
  - Therefore, **stability** is the priority.
- A system starts at a baseline of stability. A change is then made.
  - All changes have some kind of a destabilizing effect.
  - Eventually the system becomes stable again, usually through some kind of intervention. This is called the **change-instability cycle**.

Change vs Stability

- Because of the tension between the operational desire for stability and the developer desire for change, there must be mechanisms to reach a balance.
- One strategy is to prioritize work that improves stability over work that adds new features.
  - For example, bug fixes would have a higher priority than feature requests.
  - With this approach, a major release introduces many new features, the next few releases focus on fixing bugs, and then a new major release starts the cycle over again.
  - If engineering management is pressured to focus on new features and neglect bug fixes, the result is a system that slowly destabilizes until it spins out of control.
- Another strategy is to align the goals of developers and operational staff.
  - Both parties become responsible for SLA compliance as well as the velocity (rate of change) of the system.
  - Both have a component of their annual review that is tied to SLA compliance, and both have a portion tied to the on-time delivery of new features.

Site Reliability Practices

1. Hire only coders:
   - SRE might not be a full-time software developer, but he or she should be able to solve nontrivial problems by writing code
2. Have an SLA for your service.
   - Without a SLA, you don't know how you're doing and can't improve
3. Measure and report performance against the SLA.
   - Per the above
4. Use Error Budgets and gate launches on them.
   - An error budget is the maximum amount of time that a technical system can fail without contractual consequences.
5. Have a common staffing pool for SRE and Developers.
6. Have excess Ops work overflow to the Dev team.
7. Cap SRE operational load at 50 percent.
8. Share 5 percent of Ops work with the Dev team.
9. On call teams should have at least eight people at one location, or six people at each of multiple locations.
10. Aim for a maximum of two events per on call shift.
11. Do a postmortem for every event.
12. Postmortems are blameless and focus on process and technology, not people.

Operations at Scale

- **Operations** in distributed computing is operations at a large scale. Distributed computing involves hundreds and often thousands of computers working together. As a result, operations is different than traditional computing administration.
- Manual processes do not scale.
  - When tasks are manual, if there are twice as many tasks, there is twice as much human effort required.
  - A system that is scaling to thousands of machines, servers, or processes, therefore, becomes untenable if a process involves manually manipulating things.
  - In contrast, automation does scale. Code written once can be used thousands of times.
    - Processes that involve many machines, processes, servers, or services should be automated. This idea applies to allocating machines, configuring operating systems, installing software, and watching for trouble. Automation is not a "nice to have" but a "must have."
    - Automation is the subject of Chapter 12

Three Categories Of Things Are Not Automated

- Things that should be automated but have not been yet,
    - It takes time to create, test, and deploy automation, so there will always be things that are waiting to be automated. There is never enough time to automate everything, so we must prioritize and choose our methods wisely.
- Things that are not worth automating
    - Some things are not worth automating because they happen infrequently, they are too difficult to automate, or the process changes so often that automation is not possible. Automation is an investment in time and effort and the return on investment (ROI) does not always make automation viable.
- Human processes that can’t be automated.
    - Some tasks cannot be automated because they are human processes: maintaining your relationship with a stakeholder, managing the bidding process to make a large purchase, evaluating new technology, or negotiating within a team to assemble an oncall schedule.

Service Life Cycle

- Operations is responsible for the entire **Service Life Cycle**:
  - Launch, maintenance (both regular and emergency), upgrades, and decommissioning.
  - Each phase has unique requirements, so you'll need a strategy for managing each phase differently.
- The stages of the life cycle are:
  - **Service Launch:** Launching a service the first time. The service is brought to life, initial customers use it, and problems that were not discovered prior to the launch are discovered and remedied.
    - If we launch new services frequently, then there are probably many people doing the launches. Some will be less experienced than others.
    - In this case we should maintain a checklist to share our experience.
    - Every addition increases our **organizational memory**, the collection of knowledge within our organization, thereby making the organization smarter.
  - **Emergency Tasks:** Handling exceptional or unexpected events.
    - This includes handling outages and, more importantly, detecting and fixing conditions that precipitate outages.
  - **Nonemergency Tasks:** Performing all manual work required as part of the normally functioning system.
    - This may include periodic (weekly or monthly) maintenance tasks (for example, preparation for monthly billing events) as well as processing requests from users (for example, requests to enable the service for use by another internal service or team)

The Stages Of The Life Cycle

- **Upgrades:** Deploying new software releases and hardware platforms.
    - The better we can do this, the more aggressively the company can try new things and innovate.
    - Each new software release is built and tested before deployment.
        - Tests include system tests, done by developers, as well as user acceptance tests (UAT), done by operations.
        - UAT might include tests to verify there are no **performance regressions** (unexpected declines in performance).
    - Vulnerability assessments are done to detect security issues.
    - New hardware must go through a **hardware qualification** to test for compatibility, performance regressions, and any changes in operational processes.
- **Decommissioning:** Turning off a service. It is the opposite of a service launch
    - Removing the remaining users,
    - Turning off the service,
    - Removing references to the service from any related service configurations,
    - Giving back any resources, archiving old data,
    - Erasing or scrubbing data from any hardware before it is repurposed, sold, or disposed.
- **Project Work:** Performing tasks large enough to require the allocation of dedicated resources and planning.
    - While not directly part of the service life cycle, along the way tasks will arise that are larger than others.
    - Examples include fixing a repeating but intermittent failure, working with stakeholders on roadmaps and plans for the product's future, moving the service to a new datacenter, and scaling the service in new ways.

Operational Teams

- An operational team needs to get work done. Teams need a strategy that assures that all incoming work is received, scheduled, and completed. Rotations can occur for different staff on a day, week, or month rotational basis.
  - **Emergency Issues:** Outages, and issues that indicate a pending outage that can be prevented, and emergency requests from other teams. Usually initiated by an alert sent by the monitoring system via SMS or pager.
  - **Normal Requests:** Process work (repeatable processes that have not yet been automated), non-urgent trouble reports, informational questions, and initial consulting that results in larger projects. Usually initiated by a request ticket system.
  - **Project Work:** Small and large projects that evolve the system. Managed with whatever project management style the team selects.
- To implement our recommended strategy, all members of the team focus on project work as their main priority.
  - Project work is best done in small teams. Solo projects can damage a team by making members feel disconnected or by permitting individuals to work without constructive feedback.
- However, team members take turns being responsible for emergency issues as they arise.
  - This responsibility is called **on-call**. It is common that on-call duty and ticket duty are scheduled in a rotation
- Likewise, team members take turns being responsible for normal requests from other teams. This responsibility is called **ticket duty**.

Toil and Fix-it Days

Toil Reduction

- Toil is manual work that is particularly exhausting.
- If a team calculates the number of hours spent on toil versus normal project work, that ratio should be as low as possible.
- Management may set a threshold such that if it goes above 50 percent, the team pauses all new features and works to solve the big problems that are the source of so much toil.

Fix-It Days

- A day (or series of days) can be set aside to reduce technical debt.
- **Technical debt** is the accumulation of small unfinished amounts of work.
  - By themselves, these bits and pieces are not urgent, but the accumulation of them starts to become a problem.
  - For example, a Documentation Fix-It Day would involve everyone stopping all other work to focus on bugs related to documentation that needs to be improved.
  - Alternatively, a Fix-It Week might be declared to focus on bringing all monitoring configurations up to a particular standard.

Virtual Office

- Many operations teams work from home rather than an office.
- Since work is virtual, with remote hands touching hardware when needed, we can work from anywhere.
  - Therefore, it is common to work from anywhere.
  - When necessary, the team meets in chat rooms or other virtual meeting spaces rather than physical meeting rooms.
  - When teams work this way, communication must be more intentional because you don’t just happen to see each other in the office.
- It is good to have a policy that anyone who is not working from the office takes responsibility for staying in touch with the team.
  - They should clearly and periodically communicate their status.
  - In turn, the entire team should take responsibility for making sure remote workers do not feel isolated.
  - Everyone should know what their team members are working on and take the time to include everyone in discussions. There are many tools that can help achieve this.

Communication Mechanisms and Policies

- Chat rooms are commonly used for staying in touch throughout the day.
- Chat room transcripts should be stored and accessible so people can read what they may have missed.
- Higher-bandwidth communication systems (zoom, etc.) include voice and video systems as well as screen sharing applications.
- The higher the bandwidth, the better the fidelity of communication that can be achieved.
- The communication medium with the highest fidelity is the in-person meeting.
- Virtual teams greatly benefit from periodic in-person meetings.
- Everyone travels to the same place for a few days of meetings that focus on long-term planning, team building, and other issues that cannot be solved online.
- Many teams establish a communication agreement that clarifies which methods will be used in which situations.
  - For example, a common agreement is that chat rooms will be the primary communication channel but only for ephemeral discussions.
  - If a decision is made in the chat room or an announcement needs to be made, it will be broadcast via email.
  - Email is for information that needs to carry across oncall shifts or day boundaries.
  - Announcements with lasting effects, such as major policies or design decisions, need to be recorded in the team wiki / website or other document system (and the creation of said document needs to be announced via email).
  - Establishing this chat–email–document paradigm can go a long way in reducing communication problems.

Questions