Practical Monitoring

1. Monitoring Principles

1.1 Monitoring Anti-patterns

> An anti-pattern is something that looks like a good idea, but which backfires badly when applied.

Anti-Pattern #1: Tool Obsession

> When the mission is defined by running software, analysts become captive to the features and limitations of their tools. Analysts who think in terms of what they need in order to accomplish their mission will seek tools to meet those needs, and keep looking if their requirements aren't met.

Monitoring Is Multiple Complex Problems Under One Name You should use specific tools for specific purposes, not a generalist tool for specific purposes.

You should be using a configuration management tool. Agent-less monitoring is extremely inflexible.

You should use as few tools as needed to get the job done.

Adopting tools and procedures of more successful teams will not make you successful, as those tools haven't made them successful either. They use those tools and procedures because they are successful, not the other way around.

Chose your tools with care. Tools are created by teams with different goals in mind, be sure those tools fit your teams goals also.

Sometimes, you really have to build it Create small, specific tools, not big, generalist platforms.

Sometimes, the initial time invested in developing specialized tools for your specific purpose pays off in the future. An example of such a tool could be the creation of AWS EC2 instances with all your company standards automatically applied.

The Single Pane Of Glass Myth Having multiple tools feeding into a single dashboard can be ineffective. Having multiple tools feeding into their specific dashboards and then connecting the dashboards together might be a better solution.

Anti-pattern #2: Monitoring-as-a-job

Don't create specific jobs out of system monitoring. Monitoring is a skill all team members should have, and it's a task everyone in the team should participate in.

Anti-pattern #3: Checkbox Monitoring

Common anti-pattern smells:

- Recording metrics like system load, CPU usage, and memory utilization, but the service still goes down without your knowing why
- Constantly ignoring alerts, as they are false alarms more often that not.
- Checking system for metrics every 5 minutes or less
- Aren't storing historical metric data

How to remediate this:

What does "Working" really mean? Monitor that. Talk to the service app owner, to understand what "working" really means. For example, when monitoring a WebApp, check the `GET` response, latency and text to be displayed. If one of these three requirements fails, we know something has gone south.

OS metrics aren't really useful for alerting. Focus on what "working" means, instead of OS statistics. If a MySQL server is taking up all of the CPU usage, but response times are on point, then, there isn't really a problem.

Collect tour metrics more often. Monitoring often is preferred because in complex systems, a lot can change in a small period of time. Common, modern hardware can perform very well, even with more consecutive monitoring. Also, test the monitoring approach in a lab before deploying to prevent system failure due to monitoring.

Anti-pattern #4: Using Monitoring as a Crutch

Fix the problem rather than monitoring it. More monitoring doesn't fix a broken system, and it's not an improvement in your situation.

Anti-pattern #5: Manual Configuration

Automation is key. If you take a long time to add a new server to your monitoring solution, the most probable is that you will make a mistake of even don't add it at all. If, in the other hand, it takes only a couple of minutes, you can ensure the server has been correctly added to the monitoring stack and will add it.

1.2 Monitoring Design Patterns

Pattern #1: Composable Monitoring

> Use multiple specialized tools and couple them loosely together, forming a monitoring platform.

Composable monitoring can be thought of as the UNIX philosophy in action.

Components of a Monitoring Service Every Monitoring Service has to contain these 5 services.

1. Data collection
2. Data storage
3. Visualization
4. Analytics and reporting
5. Alerting

Data collection

Pattern #2: Monitoring from the user perspective

Pattern #3: Buy, not build

Pattern #4: Continual Improvement

1.3 Alerts, On-Call, and incident Management

1.4 Statistics Primer

2. Monitoring Tactics