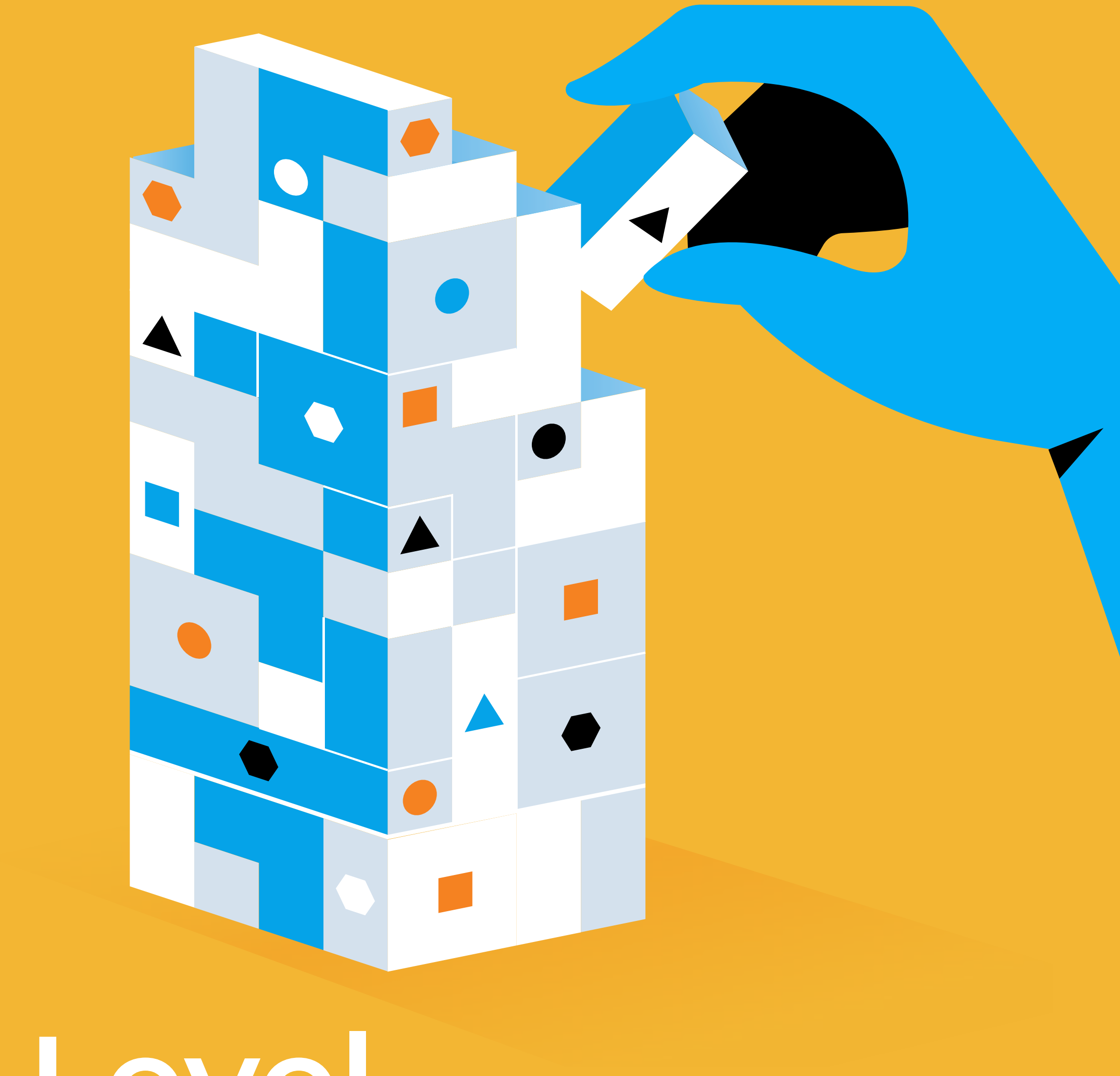


DevOps At Scale: Chapter 4

DevOps Maturity Path: Check Your Level

PLUTORA

For many organizations, DevOps maturity is an opaque concept. Someone sets up a Jenkins or CircleCI pipeline, and they say that they're "doing DevOps." These organizations then struggle to know which steps to take to grow their DevOps culture.

The problem with their definition is that it's binary, and it's simplistic. If you have a continuous integration pipeline, you're a DevOps organization. If you don't, you aren't.

Instead of approaching DevOps from a yes/no perspective, it's far better to treat it like a living organism. DevOps is about adopting a mindset of continuous improvement. The maturity of a DevOps organization is another place where that mindset must take hold.

There are a number of facets common to every mature DevOps culture. By naming and understanding them, it's possible to identify areas where a business's culture is strong and areas where that same business is weak.

The Facets of DevOps Maturity

In this paper, we'll talk about seven different ways that organizations mature in their DevOps culture. It's important to remember that every organization—and every part of the organization—won't mature at the same speed through each of these facets. However, by identifying those facets, it's possible to recognize the area where an organization is falling short and begin to shore up those weaknesses. It's also an opportunity to reinforce strengths and use those strengths to lead the organization to new heights. Those facets are

- Collaborations between teams
- Automated configuration management
- Release management
- Continuous integration
- Product mindset
- Compliance difficulty
- Continuous improvement mindset

The rest of this article will look at each of those facets at four defined maturity levels. Each level will have signposts that will help an organization recognize if they're at that maturity level, as well as steps to take to move the organization to the next level.

Level 1: Testing the Water

Level 1 of DevOps maturity is for teams who are just beginning to test the waters of DevOps. These teams are very immature when it comes to DevOps. That doesn't mean that they're immature engineering organizations. Instead, their processes are usually static and familiar, but they might not be serving the organization well. Teams at this level will regularly experience projects that go way over time and budget. When they sit down to try to figure out what went wrong, they'll make a huge list of things they've learned. Then they'll proceed to repeat those same mistakes on the next project—and the one after that.

How to Recognize If You're at This Level

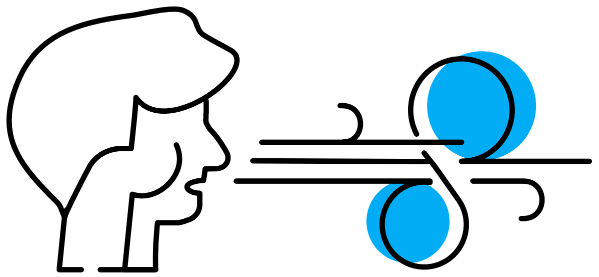
This level is where the hypothetical team that "does DevOps" by installing a Jenkins server lives. Teams at this level often times see operations as their own team, distinct from engineering or project management teams. They're rarely consulting during the planning or early implementation stages of the project. Instead, they receive new code from developers or QA with little knowledge of how it works or how it to deploy it. Then they're on the hook for trying to fit it into the rest of the system. The result is a heavily manual integration process. An operations employee might need to touch dozens of individual servers to make sure they work with the new code.

This kind of process means that there's no consistency to the configuration of important servers. That's because they were all set up manually. The team has a Jenkins server, but they're certainly not using it for true continuous integration. Instead, they automate a few build steps and perform the rest by hand. That kind of piecemeal approach leads to individual scripts for every little process the team needs to complete. Scripts like those tend to quickly become unwieldy, and rapidly become completely unmanageable. What's more, the way that the team manages projects can introduce problems for the organization. Most organizations at this step still use a traditional waterfall project management structure, in which they think of each project as its own discrete journey that doesn't carry any improvement to the rest of the business.While there is certainly still a place for waterfall project management, teams that focus on projects over the products they're meant to serve struggle to improve those products in meaningful ways for their customers. They plan everything, then code all of it, then go through painful rounds of QA and compliance approvals before the code is ready to go to the operations team. Many times, they'll do all that only to find that operations needs the code changed again.

What to Do If You're at This Level

A team at this level should look at each facet of DevOps maturity and seek to improve incrementally. The best place to start is to recognize the team's strengths and weaknesses as it pertains to continuous improvement. By adopting a more focused attitude and structured process for continuous improvement, teams will recognize that they can improve each of the other facets incrementally and independently.

From there, the answers start to become clearer on how to mature in other facets. If the operations team is too siloed, the engineering and project management teams find ways to break down those walls little by little, involving them earlier in the process. Operations can begin to adopt and standardize server configuration through configuration management tools. Engineering teams can begin to add automated tests to validate the quality of each software build. The project management team begins to shift their focus from undertaking big, challenging projects to thinking about the products their team supports and the best ways to improve them as a whole. Engineering teams involve compliance and QA organizations much earlier in the SDLC.



Level 2: Holding Your Breath

A team at this level has largely committed to their DevOps journey but may not be seeing promised returns yet. Some changes have certainly improved things for the team, but some feel like a lot of busy work for little gain.

How to Recognize If You're at This Level

Teams at this level have broadly adopted automated configuration management, and they're feeling the benefits. Automated software provisions and enforces configuration for each server. Operations staff and engineering staff regularly converse about upcoming feature code and bug fixes. New deployments don't take the operations team by surprise. They're able to plan out what configuration changes code will need, and they implement those changes while engineers are developing the feature. Freed from the necessity of always being reactive, the operations team can start to collect some meaningful data about the performance of new features. They can say with certainty which features are introducing the most bugs, how many people are using new code, and where the highest rates of return are localized.

While that data is valuable, most members of the team don't really know what to do with it yet. Moreover, there are still some hiccups. It's likely that the project management office still thinks of software releases as big projects. Groups of disparate, unrelated features are bundled together into big projects because releases are still a major event and customers won't wait for another release. The concept of a minimum viable release is still foreign, and the result continues to be lengthy quality assurance and compliance timelines. While those teams are a part of the planning and design conversations, they're not fully integrated. This means that QA and compliance still takes a significant amount of the time between when code is written and when it's deployed.

At this point, the team probably has a real continuous integration system, and it works—mostly. Operations staff likely still needs to manually intervene on a regular basis.

What to Do If You're at This Level

One of the biggest enemies at this level is complacency. Many teams will reach this level after months or years of progress and simply stagnate. They've created a process that "works for them" and lack people with the vision or political power to spur them onto more advanced steps.

Much like the fixes at level 1, the best way out of level 2 is through constant incremental improvement. Now that they've started collecting metrics about their team and software performance, teams should critically evaluate those metrics to see which are working well and discard those that don't. Operations teams should be constantly identifying new ways to automate troublesome manual steps in the deployment process. Engineers should continue to create more and better automated tests. These tests give both the engineering and QA teams more confidence that code does what it says and doesn't break anything.

Project management teams should continue to refine their processes to focus on releasing the minimum viable product for each release. Along with changes to the operations, QA, and compliance timelines, the release cadence should speed up so that more, smaller releases are happening more frequently.

Level 3: Diving in Head First

A team at this level has made it over the hump of level 2. Continuous improvement is a company cornerstone, and employees in every part of the engineering organization regularly identify new areas for improvement.

How to Recognize if You're at This Level

Teams at this level devote themselves to continuous improvement. They fanatically measure how their changes impact the business bottom line. Also, they have outstanding metrics that allow them to quantify the impact individual releases have on the overall performance of the software. Each team can reliably point to which feature introduced individual bugs. Other metrics help identify which new features slowed down (or sped up!) server performance. The deployment process is nearly automated, but it might require one or two manual interventions to make sure they go smoothly. The project management team works closely with developers, operations, and compliance teams when planning improvements to the product. The compliance organization is directly involved with code reviews so that they can identify concerns while the code is written. Your continuous integration system works perfectly well over 90 percent of the time. A broad suite of high-quality automated tests drastically shortens the QA window. Fewer bugs are written, and teams are confident new features do what they're supposed to.

Things aren't perfect here, though. It's likely that there are still some fights about what should go into a feature or release. Project management still approaches a code release as a discrete event instead of a series of continuous, incremental software changes. This means that there's difficulty knowing what should or shouldn't go into a particular feature. The company may also lack sufficient data from customers to know how to make those decisions without relying on gut feelings or guesses.

Being at this level can also lead to a feeling of frustration, as technical teams have far more metric data than management. That data might be difficult to access or challenging for management to understand, meaning that they make decisions organizational telemetry suggests will be worse for the business.

What to Do if You're at This Level

Once again, the process for moving past this level is continuous, incremental improvement. The next step for project teams past this point is to begin to unite data from the operations team directly to conversations with customers. In this way, they can identify the minimum viable product for each feature. Those metrics should also become a direct part of the decision-making portfolio for upper management, meaning that they can make decisions with effective data to support their thinking.

Ideally, teams at this level start to involve compliance teams directly in the planning process. Insecure and non-compliant code never makes it into the software at all. The operations team continues to work to fully automate their continuous integration pipeline, ironing out every need for manual intervention.

In short, changes at this level are ones of refinement, not tectonic shifts.

Level 4: Improving Lap Times

A team at this level has fully embraced the DevOps culture.

How to Recognize if You're at This Level

A team at this level has integrated continuous improvement and the measurement of performance directly into their DNA. Engineering is able to accurately say how many bugs they're introducing and what impact new code has on any environment. That data is directly tied to customer satisfaction levels, and the compliance organization has extensive input into every decision made by technical teams. Both operations and management staff are able to use hard numbers to describe the risk of adding some new feature or delaying a bug fix. How teams describe those risks can vary. Sometimes they'll discuss downtime or customer satisfaction metrics. Sometimes it's server performance. Whatever the metric, everyone involved in the process understands the data and the risk around that decision.

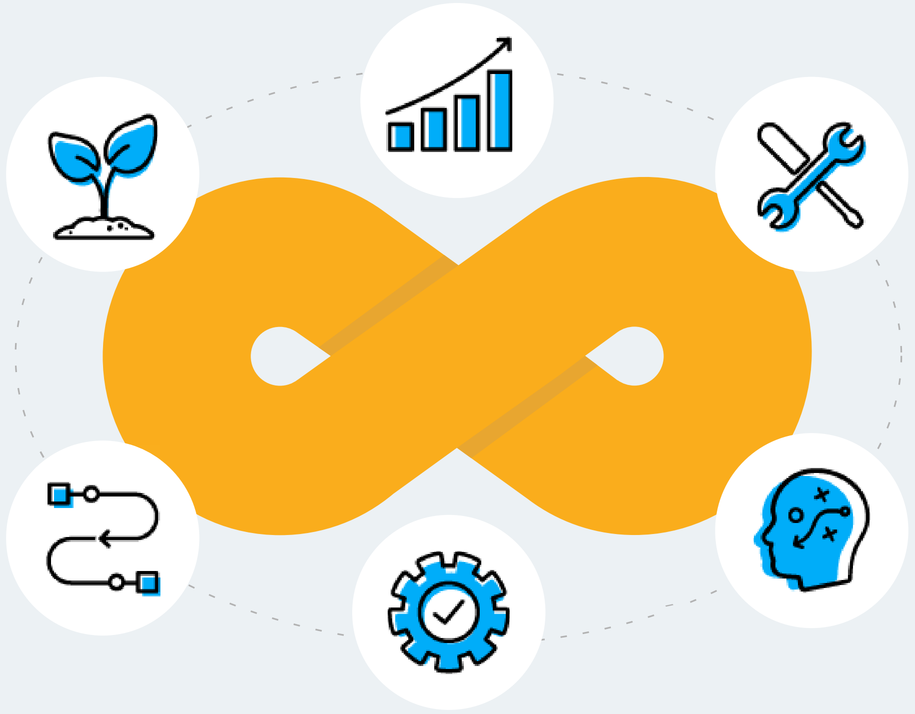
By this point, compliance and quality assurance are so built into the development process that they sign off on code shortly after it's written. An extensive, high-quality suite of tests means that deployments happen very soon after code has been finished. Organizations at this level will often deploy code multiple times per day. That's in contrast to teams at level 1, who deploy once or twice per quarter. The bedrock of DevOps, the continuous improvement mindset, is so ingrained that teams can accurately describe how they're improving. It's not just that, either; they can say by how much and over what time windows. The product team makes decisions about what features to prioritize based on hard data and conversations with key customers. In short, a team at this level is highly refined. Their process is well-defined, and everyone understands not only their role but also which steps to take to improve their performance in that role. Every day is a new opportunity to do things a little bit better.

Maturity Is Understanding That You Can’t Do it All at Once

These different levels seem clearly delineated, but they’re not. In reality, most teams don’t recognize the steps that led them from one level to another. More likely is that some series of changes pushes them to a new level and they only recognize the transition in hindsight. Immature teams will approach this process by trying to make a dozen changes at once. More often than not, they find that this means they fall flat on their faces.

Mature teams approach moving through these levels as a process. Again, the heart of DevOps is continuously improving a team’s performance in a variety of ways. Instead of attempting to take a giant step, mature teams take many little ones.

That’s the key trait of DevOps maturity: the ability to recognize places where a team is falling short and identify small changes to make over time in order to fix those things. Everything else flows from that.



Want to learn more about DevOps?

Check our series of white papers about DevOps to learn
from the foundations, to the cultural change and maturity model.

1. What is DevOps?
2. The Benefits of DevOps Strategy
3. DevOps Methodology: Aligning your Organization
4. DevOps Tools: Why You Need Them
5. The DevOps Maturity Model
6. DevOps Pipeline: The Functional Building Blocks
7. Mastering the DevOps Process
8. Making your DevOps + Agile Transformation a Success

Visit www.plutora.com/devops to learn more.

About Plutora

PLUTORA

Learn more: www.plutora.com
Email: contact@plutora.com