DEVOPS MATURITY MODEL

To assess the maturity level of a variety of critical DevOps subjects across several domains.

Your firm is almost certainly undergoing a DevOps shift. Perhaps you're having difficulty getting there. The following are excellent questions:
Where did you begin, what went well, and what may be improved?

The most critical question is, "*Where are you at this point in your journey?*"

Several organizations may be grappling with the same issues. It's challenging to find universally applicable solutions since a DevOps transformation touches all divisions and all businesses are unique. Can a framework be of assistance here? Which maturity level do you belong to? Introduce yourself to the DevOps Maturity Model.

Do you need it?

Before delving into the nuances of the Maturity Model, let's address the primary question: do you even need it?

If you are a brand new company that is cloud-native and has no heritage and capable of rapidly delivering trustworthy software from source code to production without many mistakes, security concerns, or other difficulties, you are in excellent condition. Even better if you've already begun collecting continuous feedback to enhance the next iteration. This demonstrates that you have automated a significant number of phases in the software development lifecycle. When you reach this maturity level, things seem to be hopeful.

However, what about huge enterprises with legacy code, lengthy approval procedures, segregated teams, and a high volume of manual handoffs? Businesses respond "ad hoc" to external changes and interruptions to their manufacturing environments. Consider a scenario in which a large number of developer teams lack expertise with CI/CD, DevOps, and production monitoring. Then you will unquestionably gain from charting your journey utilizing a DevOps Maturity Model.

What exactly is DevOps maturity model?

A DevOps Maturity Model is a conceptual model comprised of a matrix of maturity levels on the one hand and areas (with subtopics) on the other. Organizations use these models to assess the present (and intended) degree of DevOps-related subjects. DevOps is not limited to implementing a Continuous Integration / Continuous Delivery pipeline for your apps and infrastructure. It is far more significant in scope and does not limit itself to technical considerations. Typically, organizations using DevOps have difficulties with the following:

- **How to automate the appropriate tasks to eliminate human labor and process stages ?**
- **How to handle releases effectively ?**
- **Adopt a product-centered mentality across teams and management.**
- **How to form multidisciplinary teams that take ownership of their work ?**
- **How to build and enhance team cooperation ?**
- **How to actualize continuous integration / continuous delivery ?**

The DevOps Maturity Model serves as a "conceptual guide" for determining the maturity level of these issues and outlining a roadmap for progressing from one level to the next. Several DevOps consulting organizations may assist you in your DevOps journey.

I will outline the most prevalent themes that many Maturity Models share in the next part.

The four major Categories

Almost every DevOps Maturity Model is built on a single area of focus. These sections are representative of the many facets of DevOps inside a business.

Technology

Technology is likely the most prevalent sector, yet it is also the most demanding. Consider the following themes in this area: environment provisioning, the progression of data(base) modifications from one stage to the next, the deployment of centralized monitoring and logging, and so on.

Change Management

Who has the authority to decide what, how many people are engaged in decision-making, and how long choices take. How you do continuous testing and how you produce and maintain documentation. And, perhaps most significantly, how all procedures are arranged within the framework of all development teams.

People

This topic is entirely concerned with the human component. Nothing else is as lively and varied as this. It's about team organization (and positioning), team learning (new skills), and competency acquisition. And what about self-organizational levels?

Culture

Culture is all about team communication, whether inside the team or through the apps being provided. It's also about cross-departmental cooperation with a variety of stakeholders. However, it is also about how teams manage outages (blameless retros or not).

Each of these domains encompasses a broader range of subjects than those mentioned below. This is a high-level summary to underline that there are several facets, not just technical ones. The technical ones account for less than half of the total. Let us now examine the many stages of an organization's journey.

Levels of Maturity

The majority of DevOps Maturity Models define five maturity levels. The following are the levels.

INITIAL STAGE

At first, the company may be unaware of what DevOps is or of the possible advantages. At this level, the organization must begin from scratch. At this level, organizations often use a waterfall project management method for their IT projects, have lengthy approval and modification procedures, and build their teams around skill sets rather than products. They plan and design everything in advance of the development teams getting started, and when the development teams are finished, separate teams deploy the application to production. Tests are conducted at a very late stage of the procedure. It is quite possible that they do not practice shift-left security ideas.

Operations is a distinct crew that sits back and waits for developers to give over their apps with (in the best-case scenario) extensive documentation detailing how to deploy them. Frequently, the transfer of the new version and accompanying deployment information is marred by missing information, misunderstanding, and finger-pointing when things go wrong.

Many, if not all, of the actions required to deploy an application, are performed manually using different scripts and procedures. These scripts are cumbersome since they vary according to program or even version. Applications accumulate significant technical debt and do not adhere to common architectural principles, making it challenging to automate standard CI/CD operations. Overall, each application resembles a snowfall composed of several individual snowflakes.

The DevOps Maturity Model might be intimidating for enterprises like these. However, they may also reap the most significant benefits. It helps if someone takes the initiative and assumes a leadership position to initiate the process. Often, the initial steps are done by engineers on a grass-roots, bottom-up basis.

THE LEVEL THAT IS REPEATED

At this level, organizations understand the fundamental concepts of DevOps and use them as effectively as possible in their everyday operations. Environments and their associated settings are versioned and easily configured consistently. They are on the right track to facilitate development and operational cooperation. Changes are not unexpected but are carefully explained. At this level, organizations are not just "reactive" to everything that crosses their way. They are proactive and seek to establish repeatable procedures in the areas they are familiar with.

However, teams often provide quite large, challenging features to manage and test. Monoliths remain challenging to decompose into smaller microservices. Project managers continue to think in the context of large-scale projects that are sometimes handled from start to finish. When anything goes wrong in production, operations people must manually act. Firefighting is a pretty regular occurrence. Compliance is a constantly evolving subject that requires continual attention.

SPECIFIED LEVEL

Consistency across sectors and themes is a critical quality of this level—repeatable but standardized processes. Consider the following:

- Database updates are conducted automatically with each release.
- Non-production deployments are automatically pushed out.
- Monitoring is incorporated into each application.

Integration tests are run in the background and serve as quality gates for subsequent stages of the delivery pipeline.

Teams are structured around specific projects or goods rather than based on skill sets. Development teams strive to ensure that explicit requirements are implemented to result in apparent commercial value. Consistently authored user stories & correct usage of the team's Definition of Ready & Done.

This is consistent with the company's broader goal, which is conveyed clearly to all stakeholders. Additionally, documentation and release notes are generated automatically and are verified regularly.

CONTROLLED LEVEL

At the controlled level, all environments are appropriately managed. Database modifications and rollbacks are validated with each product iteration. The delivery procedure is well-defined and repeatable. As a result, stakeholders are aware of what to anticipate and when. In production, applications are continually monitored, and (the appropriate) metrics are acquired. Teams are adept at incorporating input into their subsequent iteration.

The company uses a knowledge management platform to collect current information and supplement it with new knowledge. Mentors coach teams to propel them ahead.

Culture is not an impediment to change but rather an enabler of company objectives. Individual development teams understand how to use cultural factors to advance their products.

ENHANCED LEVEL

Massive kudos to teams operating at an optimal level. Toil is a completely automated system that undergoes continuous testing in production. They are aware of coping with issues such as overcrowded systems. The system will automatically scale or modify to accommodate this surge. Additionally, it adapts to possible difficulties such as network outages or other infrastructure breakdowns.

Two other examples: chaotic monkey tests are conducted at random, and environments and apps are robust and do not fail. Upgrades and fixes are carried out within the confines of the internal SLO. There is no need to communicate change windows to end-users. The user experience is enhanced, and teams adopt a customer-centric mindset and behavior.

Continuous delivery, as well as constant testing and performance monitoring, are all realities. The team receives real-time input on quality assurance and compliance concerns. Tickets are immediately generated with the appropriate priority. This results in the emergence of new organizational needs, which are achieved via a fully integrated set of procedures spanning all departments.

Evaluation & Where to Begin

Even if you understand what DevOps is and how to implement it, it's a good idea to do a thorough evaluation. The variety of themes makes it difficult to determine where you are and how to proceed. To further complicate matters, teams might be at varying degrees of expertise regarding various issues within the same domain. It's a matter of identifying the most significant bottlenecks and establishing the appropriate priorities.

For instance, team A is at the "repeated level" of test automation but only at the beginning of continuous deployments. Perhaps they have a staff of really talented testers. Now, the deployment process must be accelerated to proceed to the next stage. And this is only one aspect of the IT sector.

Make it yourself...or?

One of the most challenging topics to address is how and who should evaluate. Should the team handle it, or should an external expert do it? A team may lack reliable data on its performance. An outsider specialist views the team from a unique vantage point. It's tough for a team to monitor itself while they do their daily responsibilities.

It is preferable to have an external expert assist and lead the team in determining the appropriate level of each issue. While self-assessment is a fine place to start, consulting an external expert provides a more objective perspective. A coach who has guided previous teams may also compare the current team's "score." A coach gives insights of earlier experiences and comparable companies' scenarios. All of these advantages accrue over time.

Practical Connections

Whichever route you go (do it yourself or pay an expert), it's always a good idea to do a (brief) self-assessment. For instance, the website DevOpsMaturity provides a wizard with questions to answer. The result indicates which areas you do well in and need work. You may use the result to focus your attention when hiring an external expert or create a strategy with the DevOps team(s).
Please take a moment to review the following links to current DevOps Maturity Models and other valuable resources:

- Do you practice DevOps? Essential questions to help you get started—source: DevOpsGroup's website.
- Consult the DevOps Research and Assessment group's most current report.
- Atlassian, the company behind the widely used ticketing system "Jira," also offers a DevOps maturity model.

Conclusion

To implement DevOps throughout the whole enterprise, a DevOps Maturity Model may be used to assess the maturity level of a variety of critical DevOps subjects across several domains. It is difficult to provide an accurate overview without this model due to the sheer number of variables involved. In an ever-changing environment, these facets interact or even clash with one another. Technology evolves at a breakneck pace, & it's already tricky for teams to stay up.

About ISmile Technologies

ISmile Technologies is a proud automation-enabled intelligent cloud solution and managed IT services provider, and it is your multi-cloud technology advisor & key implementation partner.

We operate globally and leverage disruptive technologies alongside deep expertise to deliver business-specific cloud solutions. We maximize impact at an unparalleled value and securely accelerate business agility while infusing competitive excellence.

https://www.ismiletechnologies.com

sales@ISmileTechnologies.com

501 S Weber Rd Unit 108, Bolingbrook, IL 60490