DevOps best practices checklist

Visit **ten10.com**

Ten10

Ten10

There is a lot of change when you start getting into DevOps. Your people have gone from being deeply technical in one area (such as software engineering) to having to understand architecture, infrastructure, configuration, databases, network, monitoring, logging and a bucket of tools and technologies just to stay afloat.

It's no wonder that while this transformation is taking place, teams lose track of what is truly important to be successful and achieve the aspirations set out at the beginning of the transformation journey.

To keep you on the right track and ensure you feel the full benefit of successfully adopting DevOps, here are our ten best practices you need to follow:

1 Technical debt must be a priority

- Ensure technical debt is part of your delivery process and is equal in priority to any other task

☐

2 'Success or learn', not 'success or fail'

- Create a safe environment for failure and protect your team fiercely from critics of any failings
- Ensure that if failure happens, it is mitigated or permanently resolved immediately

☐

3 Prioritisation is more important than resourcing

- Ensure that work is prioritised and do not take on additional work without giving up other tasks or by increasing your resources to deliver the work

☐

4 Communication is key

- Create a plan that communicates to different audiences within the business and outside regularly and in several different formats

☐

5 Sharing enables independence

- Identify the tools, technology and approaches you could share with the wider business
- Add demonstrations of this to your communication plan

☐

6 Flexible frameworks are better than rigid methodologies

- Identify the principles behind the different stages of your release process and create rules behind these that enable consistency from the wider organisation
- Build your preferred tooling into the tools and services that you share with the wider organisation

☐

7 People over process, process over chaos

- Identify processes that need to be streamlined and can be automated
- Identify areas where processes do not exist and create simple pragmatic processes

☐

8 Be truly pragmatic

- When releasing changes to the business, think about the value of the change rather than the quality

☐

9 Slow is smooth, smooth is fast

- Identify areas of your workflow that are currently being completed by humans that could be transferred to a computer within a small amount of time

☐

10 Consistency is king

- Work on your agile process and refine it until you are delivering at least 80% of the work that is started as part of your sprint
- Measure your Say/Do Ratio to ensure you are delivering what you commit to

☐

Continue reading to learn why each best practice principle is so important to making your DevOps initiative a success.

1. Technical debt must be a priority

Action: Ensure technical debt is part of your delivery process and is equal in priority to any other task

Why: When planning your teams' work, allow for technical debt resolution. It cannot be an afterthought. It must be an active part of every delivery. Even if there is no technical debt to resolve, the team should be focusing this time on improvements, not on releasing low-value features.

Think of it this way: you are trying to climb a mountain and every ten steps someone gives you a 100g weight. This is your technical debt slowing you down. A little bit of technical debt is okay. It's annoying and a bit more challenging, but it is not the end of the world. But after you've taken a hundred steps, you now have 1kg. Now it's noticeable. A thousand steps later it's 10kg and it's really slowing you down. So unless you are actively taking it off, it grinds you to a halt.

Now imagine you can't stop the weight being added but instead every thousand steps you could remove 20% of that weight. Or even better, you could put that time into creating a slide so as new weight is added it just falls straight off. Eventually, by reinvesting the time into improvements, you can increase the amount of technical debt that can be handled and how quickly you can remove it again.

This hidden weight caused by technical debt forces you to focus on it. You must create a positive process that eventually guarantees that you will always improve the situation.

2. 'Success or learn', not 'success or fail'

Actions: Create a safe environment for failure and protect your team fiercely from critics of any failings

Ensure that if failure happens, it is mitigated or permanently resolved immediately

Why: Accepting failure is a key part of a DevOps journey. It allows you to be more pragmatic and to focus on the actual problems in a structured way. But by thinking of it as a failure you are stating it had no value. That is not true. It has value because you learnt what not to do. This is why we prefer "success or learn" as whatever the result, you are learning something: either something to not do or something to do better next time.

Encouraging this learning mindset is important and you can apply it in multiple ways. For example, you can encourage learning and collaboration within a team by discussing and resolving any potential learning opportunities as a group. If an engineer released a change that broke production you can, as a group (with the engineer) resolve that issue, giving them a learning experience they won't forget and they won't make that same mistake again.

The important thing here is the shift. As you, the team, the department, and the business gradually learn to embrace the 'success or learn' mindset you will spend less time on the blame game and more time on resolving the issues, educating your staff, and creating an environment where people want to be successful.

3. Prioritisation is more important than resourcing

Action: Ensure that work is prioritised and do not take on additional work without giving up other tasks or by increasing your resources to deliver the work

Why: One thing we've heard a lot is "I don't have the resources to do that." That is only true if the resources don't exist. If any resources do exist, then it's not a resourcing problem. It's a prioritisation problem.

The most common mistake across IT management is a lack of proper prioritisation. Everyone has too much work. The difference between those that succeed and those that do not is the ability to anticipate the business needs and to prioritise their work to ensure the most valuable work is being done at the right time.

This can take the form of a formal review process between the management teams and the key stakeholders or it can be done informally but it is hard to get consensus if done one-on-one.

Think of your resources as a bucket: you can add water into the bucket but it can't overflow. If it looks like adding more water will cause it to overflow then some water needs to come out first, or we agree the water waits until there is room. If this is not acceptable then you go to the common management point and ask them what water comes out, or to increase the size of the bucket. A good manager will understand this and help prioritise the work or volunteer additional resources. A bad one will ask you to do more work with the same resources while not allowing you time to improve the services to give additional capacity.

4. Communication is key

Action: Create a plan that communicates to different audiences within the business and outside regularly and in several different formats

Why: When transforming an organisation, you must think about positive messaging and clarity. It's easy after six months for people to just say "What transformation?" When you are in a steady state it is also easy for people to forget that you had to deal with several outages or ad-hoc projects and that's why their important programme was delayed.

You can mitigate this by being open and transparent, and ensuring you communicate in several different ways. If you think you are doing enough, you are probably two to three times below what you should be doing. Here is what I consider good for a team of 5-10 people in addition to status update meetings, stand-ups and other business meetings:

- **Weekly update newsletters to the development teams/key stakeholders**
- **Fortnightly lunch and learn sessions sharing what the team has been working on and what is coming up, new tools, new tech etc.**
- **Monthly meetups to share larger projects and programmes, either internal or external**
- **Quarterly hackathons with the wider organisation to share ideas and get feedback on existing project delivery**

Some of these items are focused on pushing a message whereas others create opportunities for knowledge to be shared across the business in an ad-hoc manner. What it comes down to is creating a set of different communication methods that share what is going on and encourage some debate and conversations. Hackathons can be great for this. You can present real-world business problems and articulate what the team did to resolve them while allowing for innovation from the wider company to solve the problem. It ticks a lot of boxes when it comes to communication and collaboration.

5. Sharing enables independence

Actions: Identify the tools, technology and approaches you could share with the wider business

Add demonstrations of this to your communication plan

Why: When a transformation to DevOps starts, it's very common for people to hear 'efficiency' and translate that to redundancy. While it's not what is said, it is how it's heard. This encourages people to hold onto knowledge for job security.

When DevOps is successful, there is a mindset shift from 'that's my job' to 'that's great, I can do a new job'. To help understand this, let's talk about a software release. In immature organisations, the DevOps team or operations team still manages the releases to production. There will be some loosely veiled reasons such as security, or it's too hard, or they don't understand it, but it all comes down to FUD (fear, uncertainty, and doubt). What these engineers need to understand is that by passing that responsibility over in a safe way they can spend more time learning and developing better solutions.

By sharing information, tools, and approaches with the wider organisation, they can eventually reduce the amount of work they are doing as long as they are also investing time into making the necessary improvements.

6. Flexible frameworks are better than rigid methodologies

Action: Identify the principles behind the different stages of your release process and create rules behind these that enable consistency from the wider organisation

Build your preferred tooling into the tools and services that you share with the wider organisation

Why: If you've ever been told how to do something better, you know how annoying it is to have someone tell you to work differently. People do not like being told what to do. They like to think and create innovative unique solutions. As humans, we often ignore processes or avoid following a methodology. That is why it is better to have flexible frameworks rather than strict methodologies.

You can use any method of configuration management you like as long as:

- Configuration changes are immutable
- Configuration management only configures an application, starts a service, stops a service
- Configuration management does not configure infrastructure, deploy services or update a third-party service

With this method you make it clear what you want, what you don't want, and what is expected, all while encouraging people to use the standard approaches. You can also stipulate consequences if necessary, for example: if you do not use the standard deployment process, you will be audited against the following criteria...

7. People over process, process over chaos

Action: Identify processes that need to be streamlined and can be automated

Identify areas where processes do not exist and create simple pragmatic processes (ideally automated)

Why: When delivering change, it is always better to have a process that can be followed. The only thing better than a process is putting the people first and either automating the process or adapting it to make it more user-friendly.

Once someone has been following a process for a while they start to skip steps. There's a reason for this: humans are terrible at doing as they are told. They want some independence and to apply some of their own thought processes. This is the reason why processes are both great and terrible. If the process is too long or too complicated, no one will remember it. If they must follow a long process, they will skip steps. The trick is to not have a strict process but to have consistency in the process.

If you absolutely must have a process and it is a well-understood one, then automate it and allow your people to do anything else than be tied to a mundane, repeatable process. If you do not have a well-understood process yet, implement the most basic version of that process as possible (ideally fewer than five steps, no more than one action per step). If you can get it this simple, you can automate it.

If it is not that simple, then automate the parts you can until the process is five steps, and then automate the whole thing. The key here is that processes are best left to machines. They enjoy doing the same task multiple times and in exactly the same way. Do not waste your highly-intelligent humans on tasks that waste their time and that they are bad at. Instead, get them to automate it.

With that said, it is important to have someone who comprehensively understands the process before automating it and they should document it, but the job is not done until the process is simplified as much as possible and automated. At least this way if you run out of time to automate it you at least have a process to follow and the primary focus should be to remove the process.

8. Be truly pragmatic

Action: When releasing changes to the business, think about the value of the change rather than the quality

Why: Allowing value to be realised by the business is more important than having the perfect solution so you should identify the minimum change necessary to deliver the value, anything beyond this is a bonus.

It is not uncommon for engineers to be perfectionists. At the end of the day, we are all worried that the quality of our work and how robust it is will reflect our capabilities. With this in mind, it is hard for people to be truly pragmatic, but it enables a much higher throughput from the team if you can embrace that you are not doing a bad job, you are doing what needs to be done and if you create any technical debt you know it will get resolved.

It is very frustrating for people to learn to be pragmatic and then find the technical debt does not get resolved. This is why it must be a priority. You are allowing people to do lower quality work in the interest of speed because you have a 'success or learn' culture and a safety net of a well-managed technical debt backlog that gets actively worked on and resolved. While it is still challenging for people, it is at least much more tenable knowing that it is expected, it is safe, and it will be resolved. This does not mean that you settle for low-quality work, but you plan to do the best work you can while allowing for time to do a sub-optimal solution if you must, to ensure you are consistently delivering.

9. Slow is smooth, smooth is fast

Action: Identify areas of your workflow that are currently being completed by humans that could be transferred to a computer within a small amount of time

Why: When it comes to racing there is a saying: slow is smooth, smooth is fast. When you're trying to be as effective and efficient as possible, you need to consider the smallest possible actions that deliver the biggest impact, hence being 'slow and smooth' in the name of doing only what really needs to be done. Once you have done that, you can improve your speed, but it must first be optimised.

The same is true in DevOps. We must be aggressive in reducing wasted time whether it is our own time, someone else's, or even a computer's time. As they say, time is money and by having the most effective and efficient processes possible, you can optimise and automate to ensure that you are delivering at a rapid pace.

10. Consistency is king

Actions: Work on your agile process and refine it until you are delivering at least 80% of the work that is started as part of your sprint

Measure your Say/Do Ratio to ensure you are delivering what you commit to

Why: Everyone loves being fast. Everyone wants to be as fast as possible when it comes to DevOps. They want to get as many changes out the door as possible. While this is great from a conversational point of view, consistency truly facilitates effective change. Being consistent means people can plan around you while being fast means they are either waiting for you or you are waiting for them (either of which is unpleasant).

If you think about 99% of businesses, speed is not that important. You do not need your bank to change how it represents your account info every hour – once a month is fine. However, to coordinate those changes in the bank where thousands of people are involved, and multiple departments are reliant on knowing when something will happen, they are forced into a waterfall style of working.

Even in smaller organisations, consistency is still key. When was the last time someone outside of an engineering department asked you "Is the work too complex to deliver this week?" Or did they ask you "How long will it take to do the work?" Whether we like it or not, humans spend a lot of time thinking about time and effort, not complexity. If you want to measure the effectiveness of agile delivery, measuring your Say:Do ratio is much more useful to most businesses and stakeholders than being able to say you managed to deliver 34 points of complexity.

If complexity as a measure works for you, great, but now you spend time predicting when something could be delivered rather than when something will be delivered. Again, this can work in smaller organisations where only two or three teams are involved, but when it's 15 or 20 and you are not getting the work into sprints when needed, you can suddenly add months of delay into a simple project.

By prioritising consistency, you increase predictability and confidence, meaning the whole business will deliver more.

Need Cloud or DevOps expertise with cost-effective and pragmatic solutions?

Schedule A Free Discovery Call

Contact Us:

T: +44 (0) 20 3130 4811

E: contact@ten10.com

W: ten10.com

Ten10