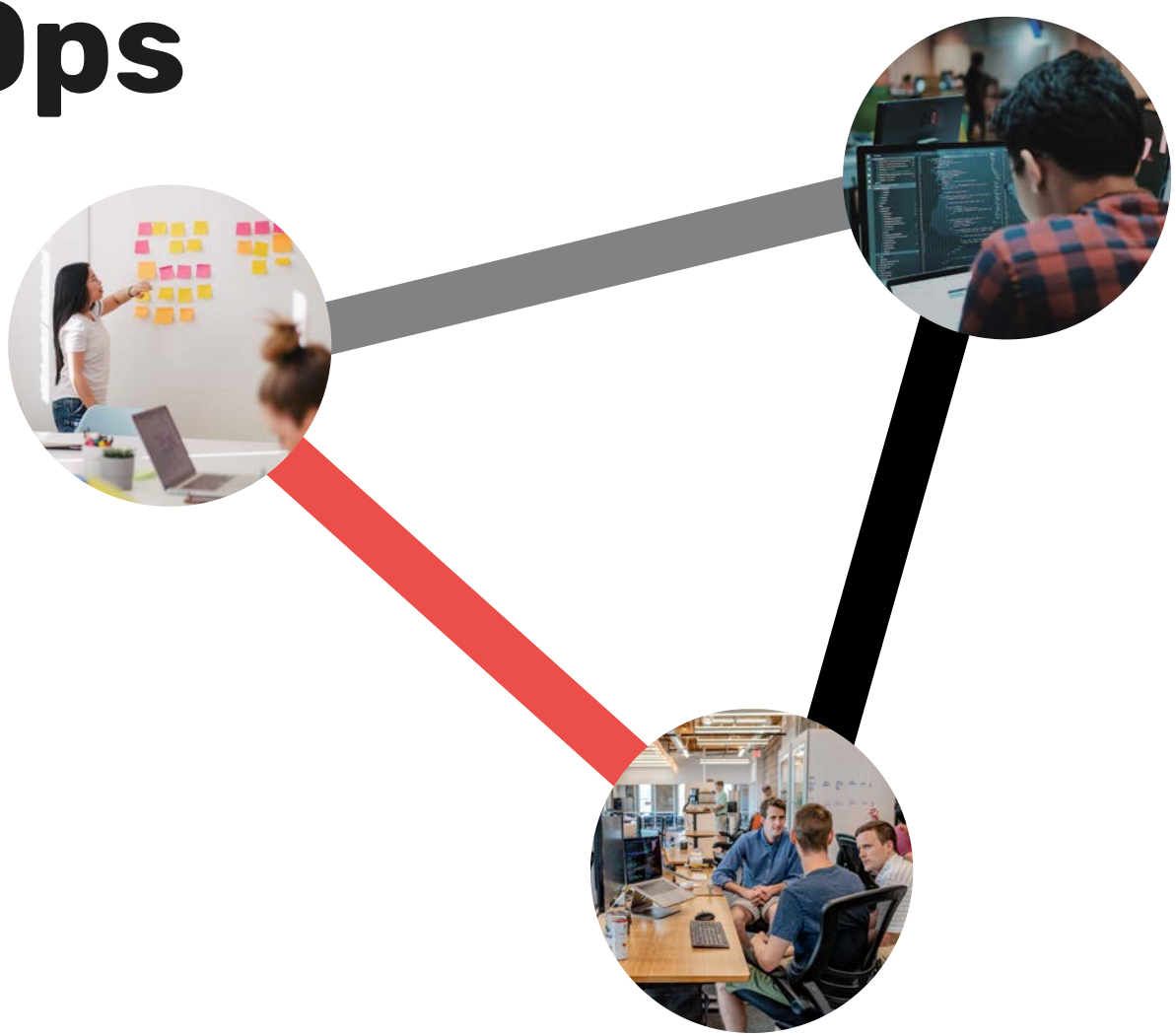
Bridging the Gap Between Dev and Ops



A practical guide to more efficient, successful projects using DevOps best practices.

nimBOLD
START WITH A FINISH

Why Should I Care About DevOps?

DevOps…You see it everywhere these days and read about it in blogs. It's a sexy term. But we've noticed there is a lot of misinformation out there. So we wanted to provide this easy-to-consume guide to DevOps to set the record straight. Similar to the term Agile Scrum, which is often touted by development shops that say "we do Agile Scrum", when it's really more like "mini waterfalling" (we call it fake Agile). And so when a company is led down a path of "fake DevOps" and the project runs into problems, many adopt the philosophy of "I tried DevOps and it failed miserably." That's really unfortunate because we get to see the power and successful outcomes of projects that implement strong DevOps practices every day here at nimBOLD.

I recently had the interior of my house painted and it struck me how seriously the professional painters took the prep process. Whereas I would certainly throw down a drop cloth over my furniture and floors, these painters took a significant amount of time to meticulously tape down plastic sheeting to every inch of my flooring, and then the careful prep they applied as they masked around windows and trim, using craft paper to cover larger areas and just basically doing most of the things I never think to do when tackling a weekend bedroom paint job with my wife. My checklist is a paint brush, a drop cloth, a roller and a can of paint. And so I don't wonder why the pros' results turn out so much better than mine.

That professional painting project is a pretty good comparison to establishing good DevOps practices. DevOps is the careful preparation, best practices, systems and the right tools that combine to propel a development project from the possibility of disorganization, unplanned wrong turns and mayhem to a more accurate, professional development experience for everyone involved. Much like the painting prep process, DevOps can look like a lot of extra time and effort up front and it can seem like an unnecessary bother to the uninformed. It often isn't until you are deep into the project that problems start showing up, milestones get missed, and cost overages set in that could easily have been avoided with the proper DevOps systems in place. But with DevOps you will see significant improvement in efficiency, cost savings and quality of the end product. We are not here to suggest that DevOps is easy. It isn't, just like true Agile Scrum isn't easy. But we promise the many benefits and rewards are well worth the extra effort.

1 Set Up an Environment of Shared Responsibility

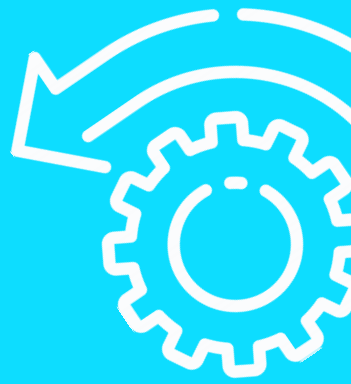
Wouldn't it be great if your development team knew the pain points of your operations team? Or what if your product team was constantly aware of the challenges your development team was facing? How would it change the way your business runs if each team was aware of each other's pain points and were able to help mitigate those issues before they even became a problem?

It has long been common knowledge that marketing and sales tend to work in silos, which is detrimental to both parties, and in turn, the company as a whole. While it has also been going on for some time, companies now need to be hyper aware of the silos that specifically affect development, because as customers demand more from technology/software in a rapidly evolving industry, collaboration and clear communication across the organization is paramount to success. We need to break down these traditional silos within software development that historically made it acceptable for teams to be disinterested in challenges outside of their circle.

Shared responsibility is a cornerstone mindset where we employ techniques to break down these silos in order to foster collaboration and clear communication between disciplines and teams. This allows teams to iterate effectively, addressing needs across all stages of development lifecycles, and ultimately improving the quality of your software.

This is not a change that will happen overnight. Depending on your company culture, there are many layers of how ingrained this type of thinking has become. Setting up an environment of shared responsibility will be a mindset and cultural shift, and it takes coaching, planning, and most importantly, discipline.

We need to break down these traditional silos within software development that historically made it acceptable for teams to be disinterested in challenges outside of their circle.



2 Enable Collaboration Through Proper Implementation of Processes and Tools

Once you have started the journey towards breaking down silos and embracing Shared Responsibility, you have taken the first steps towards effective collaboration! While people are ultimately at the center of DevOps and collaboration, processes and tools need to be put in place to strategically facilitate this collaboration.

However, these are not one-size-fits-all, which is a common mistake to avoid. Looking at another organization you think might be similar to yours and implementing the same processes and tools they have chosen will most likely not produce the same results. They need to be tailor fit to the needs of your unique organization and teams, such as your Product team, Development, Ops, QA, and more. Furthermore, they should also foster collaboration across teams, not just within one team or another.

Processes and tools are not a one-size-fits-all solution.

Some Important Questions To Consider and Evaluate Are:

- **How should we run our Agile teams?**
- **How many sprints will be involved?**
- **What project management tool will be right for us?**

Some popular choices are AzureDevOps, Jira and Trello, just to name a few

- **What project portfolio tool(s) do we need to implement?**

Is Microsoft Project or Asana our best bet? Or should we consider other options as well?

- **What source control tools will we utilize?**

Take a look at Gitlab, Github, BitBucket, AzureDevOps, and more

- **What is my release management story?**

Depending on the complexity of your software and your teams, there are different release strategies to manage your releases efficiently.

Choosing the wrong one could cause lots of pain for your teams, and cause risks to the releases of your product. However, a good strategy tailored to your team can not only mitigate these, but allow your team(s) to be more efficient.

- **What document sharing/management tool will we use?**

As your teams collaborate, you will want to be able to EASILY share and manage any artifacts that come out of discussions (meeting minutes, diagrams, process flows, etc.), so that all involved parties can easily access these documents for reference. Several tools can be used for this (Sharepoint, OneDrive, DropBox, GoogleDrive), but it is important to choose the tool that is most convenient for your team(s), while also remaining secure.

Taking the time to think through these processes and more, while also researching and comparing all of the available tools to ultimately simplify your life, may seem quite daunting at first. But you will find that it will be well worth your while to spend the time up front to gain efficiency and speed across disciplines as the project progresses.

3 Automate Your Project's Software Delivery

At this point, you should be clearly seeing a shift in your culture towards shared responsibility and deep collaboration. The Product teams are collaborating with Development teams, Development teams are collaborating with QA and Operations teams, and all teams are operating at the same cadence (Agile).

But why does it still takes so much time to see the features and software that the teams have been working on? Because you need to focus attention on automation.

Therefore, the next big step of DevOps is to start leveraging the tools you've chosen to automate as many of the manual tasks your development team executes as possible. Especially because these tasks are prone to error. Truthfully, most people already think of this step when they think of the term, 'DevOps', and they would be (partially) right. While DevOps is MUCH bigger than this, Automated Software Delivery is a very crucial part.

It's all about ensuring that as soon as any code is written and pushed to a repository, it will automatically be built and released to an environment that stake-holders can access. Gone are the days where it takes an army of coders and operations specialists hours, if not days, to ship working software. This allows your teams to focus on delivering immediate business value that stakeholders can view right away, and not get caught up in the mundane, error prone, time consuming job of building and releasing software.

Consider it the heartbeat of your software. A healthy heartbeat is consistent and continuous. However, if the heartbeat stops (i.e. build breaks, release breaks), it is an indication that something is wrong, and must be addressed immediately. Automated Software Delivery allows you to iterate and respond quickly, since you will know early and often if your software is working properly or not, and in the spirit of Shared Responsibility, allows all teams outside of development to view the software and provide feedback early.

Take comfort that it is now all automated!

4 Automated Quality

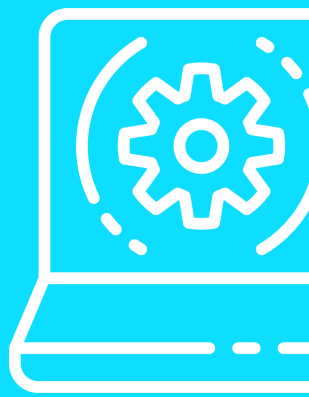
Having automated your builds and releases is a huge accomplishment in and of itself. It will allow you to iterate and deliver quickly. However, if your software is riddled with issues that negatively impact its usage, it does not matter how quickly you deliver software.

We all know that software, no matter how great it is, will always have problems. However, our obligation to our customers is to ensure that the software we provide has as little problems as possible, especially in the most critical portions of the software. Traditionally, there would be QA (Quality Assurance) teams who would exhaustively test the software after it is developed to ensure a quality release. While this is still very relevant, QA teams are still prone to human errors and depending on how large your software is, may take a lot of time, thus holding up your time before going to market.

Automated Quality is all about mitigating these barriers by introducing automation into the QA process. Before any QA team members even get their hands on the release, Automation would have automatically caught as many potential issues as possible, and the development team can start addressing the issues. This allows more test coverage of the software and quicker time to market, as issues are found early and often via automation. This also frees up your QA team to focus on other critical parts of the software and provide deeper coverage where automation may not be able to handle as easily.

DevOps strives to achieve this by putting focus on automating the quality of your software so that your feedback loop of issues is MUCH shorter, and is shared by all. Automating your quality will take time and collaboration, and is an ongoing process. There are a number of ways to automate quality, and they can be

We all know that software, no matter how great it is, will always have problems.



injected in various steps in your development cycle. Some of the more common techniques used to Automate Quality are: implementing proper unit tests, integration tests, UI automation, API Automation, performance tests, processes to properly prioritize issues, constant feedback loop. Picking which of the techniques to use and how to implement them also depends on your team and the complexity of your software system.

DevOps' Automated Quality helps navigate you through this web.

Combined with automated builds, we can now also automatically run many of these tests so that issues are automatically found earlier in the cycle, allowing your teams to respond and improve quickly and iteratively.

5 Monitoring and Health Checks

At this point, your teams should be humming right along. With quality software being delivered early and often, you are ready to release your software into the wild!

But the wild is a scary place. Regardless of all the research and testing you ran, users will potentially be doing things in your software that you could have never anticipated. You need to know when issues of varying topics in the wild arise, or what users are doing, and how it is affecting your system.

Since responsibility for quality is now shared, thanks to DevOps, teams can now collaborate to choose the proper tools to expose the appropriate metrics for the organization to ensure that the software is truly healthy. You'll want to monitor activities such as:

Traffic - How much traffic is being driven to your software? How many users do you have and how long are they using your software? Are there any peak hours where your traffic is high and you need to scale? Traffic monitoring will allow you to answer these questions and address them accordingly.

Application logs - Application logs are used internally, and allow your DevOps teams to capture information about the system that are specific to the software itself. This allows teams to easily troubleshoot issues as they arise.

Unusual activity - You will always want to know if there is activity on your software that is out of the norm. This could indicate a number of things including, issues in your software or an attack on your system. You will want to address these right away depending on the severity.

Spikes in certain calls - Similar to monitoring traffic, if there are spikes in certain calls in your application, you may want to alert your team right away to be on alert and monitor the calls in case any action needs to be taken. Spikes generally indicate that your system is being used heavily in a certain area, that may affect things like performance for your customers.

High error calls - if your software is experiencing high error rates in certain areas, it could mean disruption for your customers, which could have many unintended consequences. You will want your team to be alerted in these situations so they can run a Root Cause Analysis to see where the issue may be.

This monitoring provides yet another feedback loop where teams can now easily identify trends and issues, and start preparations to address them in future releases.

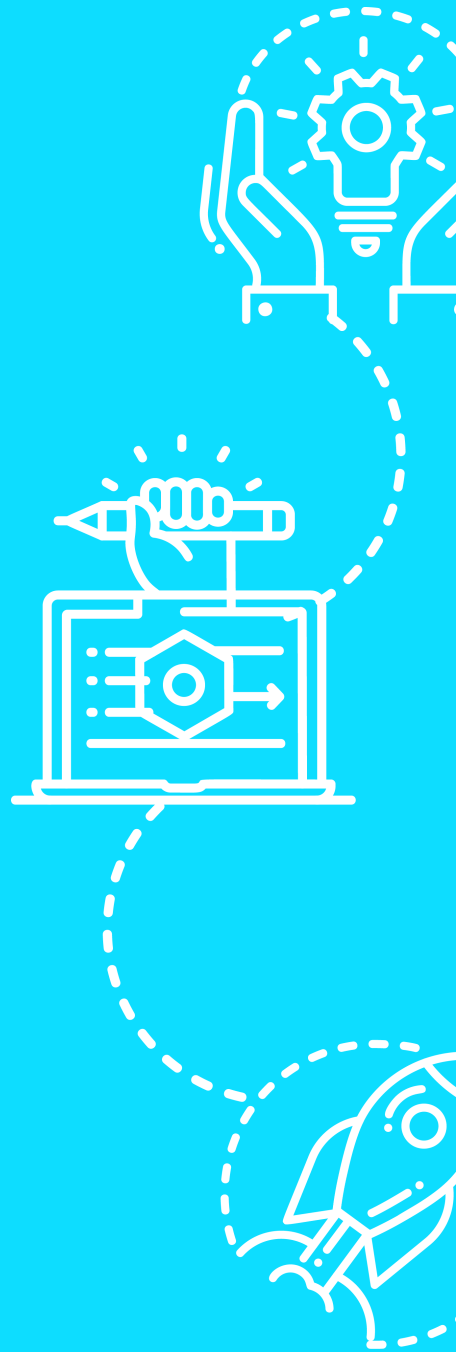
6 The Wrap Up (...and Roll Out!)

As is with maintaining your home, it never ends. These steps provide you with a framework on which to build. You now have all the fundamental pieces to deliver quality software in a timely manner within a collaborative and efficient environment. But it does not end here. DevOps is a journey, not an end destination. DevOps is a continuous loop of "rinse and repeat" throughout your development cycle, and you'll see the most success by looking for ways to improve and optimize the process each time, whether it's adding a new tool or completely changing the way you perform a set of tasks. But, if it is implemented and integrated within your culture, we are confident you will see an increase in collaboration and productivity in your teams, and will realize a faster time-to-market with better quality.

About nimBOLD

Great goals and ideas are just that—goals and ideas—until they're accomplished. Until you cross the finish line, you're just racing. nimBOLD exists to take your ambitions and turn them into accomplishments.

Our motto is "start with a finish." That means that we begin every project by clearly defining the end goal. We also believe that the shortest distance between an idea and its completion is a well-crafted strategy. Once the destination is clear, we then create a plan to ensure that every single action taken is an action towards achieving your goal. There are no superfluous tasks, trials, or talks. We live and breathe maximizing efficiency and minimizing time and cost.



If you're lacking a clearly-defined goal, we can help you uncover it.

If you're searching for a way to make progress towards that goal, we can be your catalyst.

If you want to up your game and learn how to define goals and complete complex projects like a pro, we can be your guide.

nimBOLD
START WITH A FINISH

CONTACT US

riad.bacchus@nimbold.com
(714) 931-3985