Understanding DevOps & Bridging the gap from Continuous Integration to Continuous Delivery

Manish Virmani
Development Manager IBM Rational
mvirmani@in.ibm.com,mvirmani@gmail.com

***Abstract*—As part of Agile transformation in past few years we have seen IT organizations adopting continuous integration principles in their software delivery lifecycle, which has improved the efficiency of development teams. With the time it has been realized that this optimization as part of continuous integration - alone - is just not helping to make the entire delivery lifecycle efficient or is not driving the organization efficiency. Unless all the pieces of a software delivery lifecycle work like a well oiled machine – efficiency of organization to optimize the delivery lifecycle can not be met. This is the problem which DevOps tries to address. This paper tries to cover all aspects of Devops applicable to various phases of SDLC and specifically talks about business need, ways to move from continuous integration to continuous delivery and its benefits. Continuous delivery transformation in this paper is explained with a real life case study that how infrastructure can be maintained just in form of code (IAAC). Finally this paper touches upon various considerations one must evaluate before adopting DevOps and what kind of benefits one can expect.**

Keywords—DevOps, Continuous Integration, Continuous Delivery, Infrastructure as a Code (IAAC)

I. INTRODUCTION

The way enterprises deliver software is going through a wave of change as the environment, enterprises operate in, is changing – market needs are changing continuously, technology is changing rapidly, there is more pressure to adapt to market needs and deliver quickly. Enterprises can no longer afford to make a customer keep waiting for 6 months or 1 year for a release to come and then solicit feedback from customer on how software behaves. Customers expect continuous engagement so that they can provide continuous feedback. In order to meet the challenges of today, enterprises need to be lean and agile in all the phases of software development life cycle. Over the years organizations have adopted many process optimizations in their software development (agile transformation) practices. However in this entire evolution – the focus has been mainly software development leaving the operations side of software delivery lagging behind in this optimization race [9]. As a result over the years what has happened is that software development teams are able to deliver at a much faster pace than the pace at which operations teams can absorb the builds. It is rightly said that strength of the chain is as strong as the weakest link in the chain – so is the case in software delivery i.e whatever optimizations you do in your software delivery cycle – if one phase is not able to keep pace – your entire delivery will be delayed.

DevOps is set of practices that is trying to bridge developer-operations gap at the core of things [5] but at the same time is not limited to this development and operations handoff instead covers all the aspects which help in speedy, optimized and high quality software delivery. DevOps is an set of principles towards software delivery where the key focus is on speed of delivery, continuous testing in production like environment, be in shippable state at any day, continuous feedback, ability to react to change more quickly, teams working to accomplish a goal instead of a task (no more team boundaries causing a delay[1]. DevOps extends agile principles to entire software delivery pipeline. Though DevOps principles apply to entire SDLC but key motivator or focus zone which triggered all this – is making sure operations team can run along with development teams [9]

Section II talks about DevOps applied to various SDLC phases. Section III represents the end to end DevOps enabled pipleline for software delivery. Section IV takes a deep dive into various aspects of moving from CI to CD. Section V & VI try to summarize the key points.

Fig. 1 Development Builds Flow

978-1-4673-7551-1/15/$31.00© 2015 IEEE

II. DevOps applied to various phases of software delivery

A. Continuous Planning

Businesses plans have to be agile (and they have been to an extent) i.e able to adjust quickly to the changing market conditions. Always have interim checkpoint in plan to reassess the situation and modify / adjust the plans as needed based on market feedback. It is difficult for Dev/Test teams to adapt to the quick changes in business environments. DevOps allows you to do that by always having a prioritized product backlog, continuous channel of feedback with customers and ability to prioritize the product backlog all the time, directly taking business angle in consideration. There is a continuous process to plan small portion – execute - get feedback – react to feedback and adjust plan if needed and the cycle continues.

B. Continuous Integration

Continuous integration basically refers to integrate early, don't keep changes localized to your workspace for long, instead share your changes with team and validate how code behaves continuously. Not only share within component teams but integrate beyond component boundaries, at product integration level. Further this stage of process optimization refers to achieving automation such that as soon as developer delivers the change the build systems detects that (may even be a scheduled event at end of day) and triggers a build carries out sanity tests and posts the build to a repository. This has to be a repeatable continuous process all across the development cycle.

C. Continuous Deployment

This is heart of DevOps and forms the critical piece of overall software delivery optimization. Surveys have shown that in majority of organizations the operations side of delivery is significant contributor to the delay in software delivery. Setting up the hardware to test the development build may take time varying from days to weeks. On top of that these deployment processes are manual and not consistent. DevOps principles recommend to automate the deployment and provisioning of hardware and various cloud providers play a crucial role in this field. DevOps approach proposes that entire infrastructure provisioning should be maintained as code in source code repository – concept being called Infrastructure as a code (IAAC)[6].

D. Continuous Testing

Prerequisite to continuous testing is - *Automate every test case!!* . Any process that has to be repeated over time – should get automated, there are enough technologies available to meet that goal. Manual testing process must be evaluated for possibilities of automation and in majority of cases there will be ways to automate the same. Software delivery process should be able to execute the test suite on every software build generated automatically without any user intervention thereby moving towards the ultimate goal of being able to ship a quality release quickly (why not daily !!). This whole principle of continuous testing not only moves the testing process to early in cycle but also allows the tests to be carried out on production like system (complemented by continuous deployment).

E. Continuous Monitoring

As discussed in adoption approaches above, with the capability to test early and on a production like system there is an opportunity to observe various quality parameters throughout and hence ability to react to any surprises in timely manner.

III. Software Delivery Pipeline with DevOps

Figure 2 shows the delivery pipeline with various DevOps approaches playing in. It could be compared to a delivery pipeline of a manufacturing unit. Every build/release (equal to a product in mfg.) needs to go through this automated pipeline of smoke / fvt / regression / stage / production tests phases to clear all the quality parameters and if this pipeline is automated, that will help in quick and consistent releases.

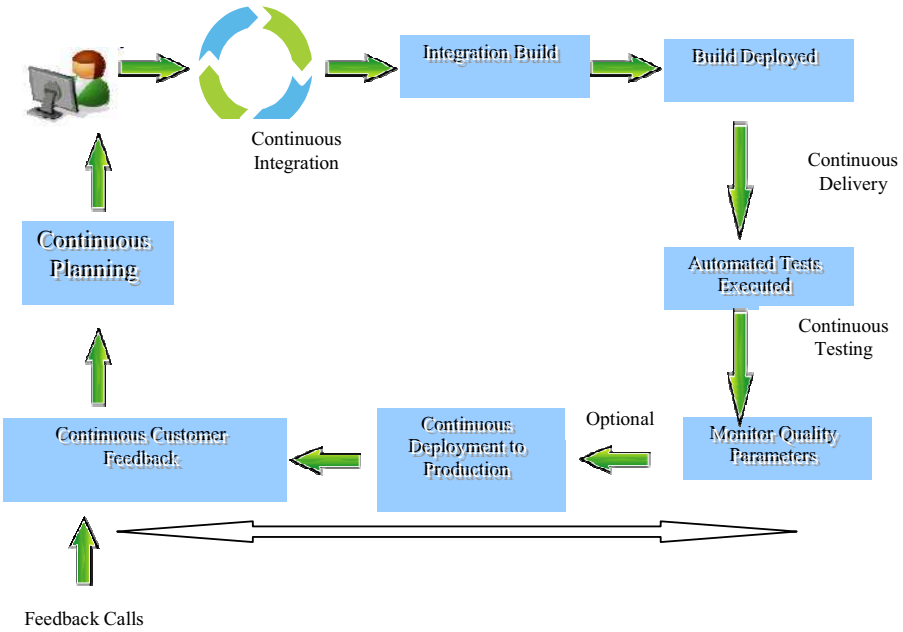


Fig. 2. DevOps in Action

IV. Bridging the Gap from Continuous Integration to Continuous Delivery

Continuous Integration processes have been here for a while now and as part of transformation over last decade most of the project teams are having processes in place which align with continuous integration (may be partially or fully). Section below tries to highlight what is the business need now to move to CD and benefits it offers.

A. Business Need

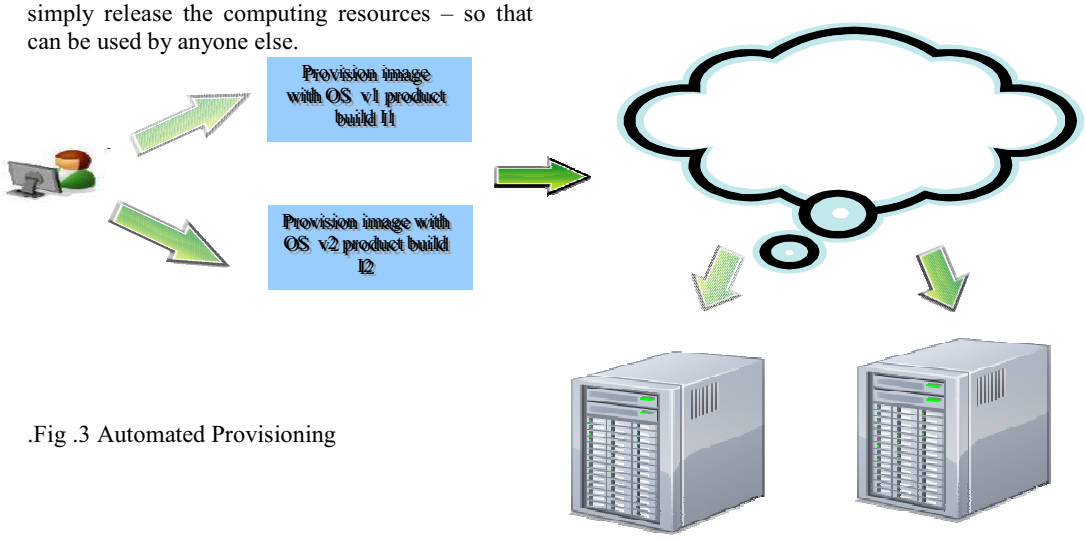
With ongoing growth of customers base and market outreach it becomes essential that organization's internal delivery processes are in line with business expectation and are optimized to best possible extent. Time and resources are critical constraints in any business environment. With these given constraints business are expected to react to market needs more quickly with high level of quality. No organizations can afford to live with manual, error prone and repeated activities in the software delivery lifecycle. Earlier the project teams identify this precise business need and adopt DevOps to optimize their processes, it is going to reap more fruits.

As explained above in this article, one of the area which generally needs attention is testing phase of SDLC as this phase has dependency on operations team for hardware provisioning, test setups creation, maintaining multiple product and operating system combinations etc to talk a few. Icing on the cake is – all of these processes require dedicated person to execute all the setup tasks manually and setting up these things are always unpredictable mostly due to unknown surprises. Teams have to manually provision these test setup, recover from crashes of dedicated hardware at the critical release junctures.

Continuous Delivery tries to optimize the infrastructure management and the critical need to balance out time and resources. Validating the software products involves multiple variable inputs like product versions, different operating systems, different third party software versions etc. It is neither humanly possible to test all these combinations nor practical. Team use various test methodologies to optimize the test matrix and keeping dedicated hardware for some key configurations – BUT this is not just working out well. Teams need to seriously look for alternative ways of maintaining test infrastructure and reducing overall FVT cycles.

B. How to bridge the gap : case study

There are two key pieces to address the business problem stated above – one being deployment automation tools (e.g IBM uDeploy) and second being cloud based resource provider (e.g IBM SmartCloud Orchestrator)[7][8]. Project teams can identify the various topologies needed to be tested and create corresponding deployment pattern [4]. Not only create deployment patterns but also automate the steps needed thereafter like - installation of application stack on the cloud provisioned image, populating the test data on this image, triggering the automated test suite and pushing the results to a central repository. Team must carefully choose the automation language here (Chef/ Puppet / Shell Script / Perl / Python etc) keeping in mind the platform coverage needed. Once this automation is built in getting to any test configuration is just pick and choose the input variable (version, os etc) and click "Go" and you are done!! Figure 3 illustrates how a user simply selects OS version, build ID and clicks on deploy which in turn provisions the virtual image in the cloud with the given specifications – all without any human intervention [6]. This automation not only helps in reducing the FVT cycles but also another very important use case is defect validation by developers on generated spins. Overall, developers are able to perform the defect validations much more quickly without having to wait to manually configure the hardware with latest software bundles having their fix in it. With this automation, developers have full control – to validate any defect – they have to just pick and choose the config and within few clicks – they will have a setup up and running on which they can validate the defect in **production like environment**. Last but not the least – once done, simply release the computing resources – so that can be used by anyone else.



.Fig .3 Automated Provisioning

C. Benefits

- **Time Saving:** Time to get any test setup is just a matter of few minutes of user action compared to days without this automation.
- **No need for Dedicated h/w:** There is no longer a need to keep dedicated hardware as after validation cloud resources can be released to be used by others thereby leading to better sharing of resources.
- **Reliable & Scalable infrastructure provisioning through IaaC:** As everything is maintained as code – this code is stored in version control repository – the entire automation is portable to any setup (with minimal tweaks). Since automated, the test infrastructure provisioning is now **consistently setup successfully** every time (no more weird unknown issues). Needless to say – it's scalable – limited only by cloud resources.
- Deployment becomes a consistent repeatable process
- Continuous deployment model enables developers to get access to production like systems and thereby doing validation in a environment similar to production
- Significant cost saving as teams can share the pool of resources. As deployment is automated in almost no time there is no need to keep resources reserved – as soon as done with testing, team can release the resources

V. APPLICABILITY

Fundamental problems that DevOps approach tries to address are adaptability to change, speed to market and maintaining high quality with low cost – which are universal business problems in any type of software project. DevOps principles are generically applicable to software delivery and are not tied to any specific type of product or services. They can be applied to enterprise level complex product development or to a small web application or even to a mobile app development. In addition to this, it really depends on project needs, organization needs, organization capabilities, return on investment which will determine what all approaches or principles of DevOps an organization should adopt or can adopt. It is not necessary (nor might be practical) to move from no-DevOps state to full end to end DevOps state in a short time [3]. Organizations or for that matter project teams should analyze and asses the current workflows which are in place and figure out the areas where there is a scope for optimization and which will give highest ROI and just target that phase in isolation (could be just 1 or 2 approaches of DevOps).

VI. CONCLUSION

As explained above in this paper how various principles of Devops if adopted in release lifecycle can really optimize the organization's capability of software delivery. DevOps not only involves change to processes but also change to culture. Organizations who are adopting DevOps principles will definitely have an edge over the organizations which are not riding this wave of DevOps. Processes have to change with time as the market environment we operate in is continuously changing. DevOps enables organization to

- Reduce time to market
- Adapt to continuous feedback
- Effectively balance out cost and quality
- Have more predictability in releases
- Increase organization's efficiency as whole

DevOps just defines the set of principles but *"how and using what technology"* organizations adopt that approach or principle is completely to be evaluated and decided by the organization. Even within a single organization different teams might have need to adopt different technology or tools to adopt DevOps approaches – which is absolutely fine, whole purpose being to continuously optimize and transform.

REFERENCES

1. DevOps for Dummies – by Sanjeev Sharma, https://www14.software.ibm.com/webapp/iwm/web/signup.do?source=swg-rtl-sd-wp&S_PKG=ov18162
2. Adopting the IBM DevOps approach for continuous software delivery, http://www.ibm.com/developerworks/library/d-adoption-paths/
3. Managing and deploying virtual system patterns, http://www-01.ibm.com/support/knowledgecenter/SS4KMC_2.3.0/com.ibm.sco.doc_2.3/t_manage_patterns.html
4. Defining DevOps, https://sdarchitect.wordpress.com/2012/07/24/understanding-devops-part-1-defining-devops/
5. Understanding DevOps – Infrastructure as a Code, https://sdarchitect.wordpress.com/2012/12/13/infrastructure-as-code/
6. How to do a full stack release with IBM UrbanCode Deploy and IBM SmartCloud Orchestrator, https://www.ibm.com/developerworks/community/blogs/invisiblethread/entry/how_to_do_a_full_stack_release_with_ibm_urbancode_deploy_and_ibm_smartcloud_orchestrator1?lang=en
7. UrbanCode Deploy and SmartCloud Orchestrator, http://drschrag.wordpress.com/2014/06/24/urbancode-deploy-and-smartcloud-orchestrator-part-1/
8. DevOps Distilled, The Three underlying principles, http://www.ibm.com/developerworks/library/se-devops/part1/