**DevOps Guide: How, where and when to start**

***It’s not an easy journey to become a DevOps Engineer!!***

One of the main challenges or short comings of the new breed of DevOps engineer can be summarized with the saying” *Jack of all trades, master of none.* ”

This is very true for freshers or people from both operations or development team who are trying to get into the DevOps career seeing the current demand for DevOps Engineers in the IT Sector.

I am not trying to say that it’s very difficult for someone to become a DevOps engineer. It’s just that you need to do a lot of preparation, research and hard work before you can be recognized as a good DevOps Engineer.

**Where to Begin**

Where people usually fail in their DevOps journey is that they put more effort in learning the DevOps tools and frameworks rather than understanding DevOps principles or to be aware about what DevOps itself is. I have gone through hundreds of resumes of candidates interested in DevOps job roles in various organizations and most of the time they highlight their knowledge in the tools rather than DevOps itself.

Secondly, people think DevOps is all about automation, while this is true to a certain extent, they fail to realize that tools such as Hudson, Jenkins, scripting jobs existed long before DevOps became popular.

While interviewing candidates for DevOps roles, I usually ask them this simple question.

“*Explain to me what DevOps is without any technical jargon” , or “Imagine I am from a non-technical background– Explain DevOps “*

Only a very few candidates could give a satisfactory answer. Most of the time people end up saying it’s automation, continuous integration and deployment, DevOps is a framework, etc. Even though some of the answers are true -maybe in a practical aspect, most of them don't define "What DevOps is".

Only if you can understand DevOps from a holistic point of view starting from what was the reason for DevOps to come into existence, what is the need for DevOps, challenges that development and operations teams face, etc – then only you can start to make sense of DevOps and give an apt answer to such questions.

**Where to Begin?** – Understand DevOps Principles and Why DevOps is a culture before you start focusing on the tools for DevOps. One way to begin that journey is by reading the book/novel – [The Phoenix Project](https://www.amazon.com/Phoenix-Project-DevOps-Helping-Business-ebook/dp/B00AZRBLHO/ref=as_li_ss_tl?ie=UTF8&keywords=phoenix+project&linkCode=sl1&linkId=b7c808063a5cc9680c28b0d0b298f0cf&qid=1491360863&sr=8-1&tag=miccom07-20). Even though it’s a work of fiction it helps in enabling the reader to gain more insight into the DevOps concepts in a very practical manner. Both in a business as well as technical point of view the reader will start to make sense of why both as an individual and an organization you need to adopt DevOps.

**Get into the Developers Shoes**

There’s no way around this and I am sure DevOps engineers who came from development background will agree to this. You need to be aware of what the developer is actually doing during the coding, development, testing and deployment process.

For people from operations, system administration background or freshers – I am not saying you need to learn development in depth or become an expert in development/programming. Rather, if you want to effectively implement the said DevOps practices and tools, you need to think like the developers, understand their requirements and give them the best solution.

Basic to intermediate knowledge in various programming languages and frameworks will help you in a long way. Therefore many DevOps engineers who worked as developers for few years are very successful in their career.

So, to summarize, understand what the developers do during the development process of the software. How they manage their versions of code, integrating the code into existing code base, resolving merge conflicts and how they are compiling and deploying them to the servers (even though Ops team handles this, developers also do play their part).

Once you have understood how things are done manually with no automation tools, then you can bring DevOps practices and tools more effectively into the picture.

**Become an Operations Expert before you even THINK DevOps!!**

I don’t know how much more I can put stress or make this point important for all aspiring DevOps Engineers. You need to be an expert in System Administration and Operations before you become skilled in DevOps. The IT world existed and ran successfully long time before DevOps was in the picture. And this was mainly due to superheroes named as Sysadmins and Ops guys. And these were a rare breed of engineers (now getting rarer thanks to all the automation tools) who knew various scripting languages better than their mother tongue.

They had good expertise in system engineering where they managed Linux/Windows servers, setup Web Servers, deployed artifacts into them via FTP and SSH, managed n number of servers. And they knew Shell better than their wife/girlfriend. Even before the dawn of DevOps they had expertise in building automated tasks (via Scripts) to automate deployment, builds and monitoring the server and applications. Today there is an misconception that you don’t need expertise in managing these tools, Operating Systems and Scripts for becoming a DevOps engineer. It’s not true.

You can begin this by,

* Learning Installation and Administration of various Linux Distros
* Start learning how to do scripting, learn bash, Perl, python, ruby etc (be an expert of at-least one scripting language other than bash script)
* Know how to setup Web Servers, DNS Servers, load balancer’s – Apache, Nginx
* Learn how to setup proactive monitoring for the infrastructure and software
* Deep understanding of Networking Concepts – example – IP Gateways, TCP, CID, ARP, etc
* Understand how to do deployment of various artifacts manually to the servers
* Gain expertise in various RDBMS systems such as PostgreSQL
* Expertise in file System management such as ext,ext4, NFS ext.

**Next, learn how to manage your code**

Configuration Management tools did exist long time before DevOps was popular. And one of the initial tools which were used for the same were SVN, ClearCase, Mercurial and Git for Version Control. Before I go further let me make this clear. **You need to have good command in any Distributed Version Control System if you want to call yourself as a DevOps engineer. By Distributed VCS I mean tools such as Git and Mercurial (I prefer Git – will elaborate in another article).**

In a way it was the extensive use and features that tools such as Git offered which pushed forward the growth of DevOps tools. Gone are the days where Ops team use FTP, WinSCP, rsync etc to transfer your code, artifacts and data between your servers.

Not only are these tools efficient in version controlling your code but also helps the ops team to use these tools for allowing the code and artifacts to be sent, processed, deployed using other configuration tools such as Jenkins, Chef, Puppet, etc in a more secure, faster and easy to manage matter.

So, to summarize – Become an expert in Git and learn basics of SVN, Mercurial and ClearCase (for legacy users and migration requirements).

**Let there be Jenkins (*or any other CI server for the matter* )**

As I had said earlier. Jenkins existed long time before DevOps became popular. But this tool was what gave the Ops team a huge success in enabling more stable build and release with its automation features. And for a long time, Jenkins and Hudson (Jenkins came from Hudson) ruled the Continuous Integration (CI) and Continuous Deployment (CD) realm. Presently tools such as TeamCity, Bamboo, GoCD and currently many tools are bloating the DevOps CI and CD domain. But personally, with the massive community support and plugins available, I personally feel Jenkins still is the best user friendly and reliable CI and CD solution.

So, what’s Jenkins?

It’s a Continuous Integration and Continuous Delivery solution. What CI does is to merge the code from different developers into a single repository/project multiple times a day while continuously testing the code base to avoid downstream issues. Now CD is the next step further where you will make sure that all the merged code is ready for production.

For both Developers and Operations Jenkins was a bliss. It helped to automate the process as much as possible right from pushing the code from the developer’s system to deploying it into production.

**So, put some time and effort to gain expertise in Jenkins first.** And as I said earlier, you can only use Jenkins in its full potential once you have gathered significant knowledge and expertise being a good operations/sysadmins engineer.

Again, I am not promoting Jenkins. But for new users, Jenkins is far easier to learn and manage compared to the other CI/CD tool there on the market. (Please correct me if I am wrong)

**Learn how to treat your “Infrastructure as Code” – Configuration Management**

This is where you get your hands dirty with DevOps tools. And before you go any further, let me kindly remind you again that you need to gain good expertise in all the above-mentioned tools and processes before you start learning these Configuration Management tools.

Unless you have good knowledge in the operations side (Linux/Windows Server Administration, Managing Web servers, Networking) , code version control ( git, SVN) and CI and CD ( Jenkins, Bamboo etc) you won’t be able to truly understand where Configuration Management tools (hence forth called CM) fit in.

Too be honest it was the CM tools such as Puppet, Chef, Ansible, SaltStack etc which bought the attention to DevOps to the market. What CM tools provisioned was the ability to manage your infrastructure as code. Either all the manual work you had done before to install an OS, install and configure its dependencies, configuring the various network parameters and setting up access levels, copying and deploying the artifacts and managing the hardware resources – all this was done manually which always lead to chances for failure. And imagine doing this multiple time for many different servers. It was very difficult to keep track and have a dependable infrastructure this way. This was where these tools came to the rescue.

**I would suggest beginners to start with Ansible and then learn Chef, Puppet and other CM tools depending on the requirements.**

**Proactive Monitoring in DevOps**

Continuous Monitoring existed long time before DevOps came into the picture. But it started to gain attention and more tools came into market or ***proactive* monitoring** of the environment recently. These tools monitor your logs and system resources and can identify potential bottlenecks and send alerts to prevent outages and performance issues.

Tools such as Nagios, senSU, logstash, New Relic, etc can help you get started.

**Entering the world of Virtualization (now evolving to Containerization)**

Virtualization was well known for more than a decade with tools such as VirtualBox, KVM, VMware etc. The difference DevOps brought here is the automation of setting up, configuration and management of these virtual machines. This is possible via tools such as Vagrant, Packer etc.

Another domain which is of the highest demand which helped many organizations in pushing DevOps is containers. Containers are light weight self-contained execution environments. Compared to regular virtualization solutions, containers are more efficient to run and easier to manage. Moreover, what containers have done is to encapsulate the software together with the right environment where it needs to be run with all the required dependencies. Either the developer can deploy his code into containers which already have the correct environment where the code can be tested and later sent to ops team to verify and deploy it – all in a matter of minutes or less compared to traditional methods.

Containers became popular due to Docker. There are also other tools to manage and scale containers such as Kubernetes, Mesos, etc.

**I would suggest starting your journey with Docker first before you get your hands on other tools to scale and manage containers.**

**Moving to Cloud**

Together with the above said Configuration Management tools it was the demand for moving into cloud and managing the cloud infrastructure which further increased the demand of DevOps. Having in depth expertise with various cloud providers and their services is essential for a DevOps engineer. Having good knowledge and experience with Cloud providers such as AWS, Azure, Google Cloud etc is very much required. Also gaining certifications provided by these providers will add value for your DevOps expertise.

**Summary**

Well as I had said earlier, it’s indeed a long and difficult journey in gaining expertise in DevOps and it will be more difficult for freshers and people with no operations nor development experience. But once you have started working hands-on the said tools and process and start to work on live projects where DevOps is being implemented, you will be gaining good knowledge and will be able to build your expertise in it.

Even though it’s very important to become strong in DevOps principles and culture, in the end for a person or the team implementing or bringing DevOps into practice – you need both the right tools and the correct process to be working together very well.

**10 DevOps Skills To Look for in Job Applicants**

**#1 – An Impeccable SysAdmin**

Must be a senior level Windows/Linux Administrator (Either/Or/Both depending on your shop) with 5 – 10 years of experience. Why? Because they need to be able to build and administer servers in their sleep. But that’s not the only reason, a lot is riding on someone to automate server deployments because this is a big problem in most IT shops.

**#2 – Virtualization Experience**

Must have 3 – 5 years of virtualization experience with VMware, KVM, Xen, Hyper-V, or whichever favor hypervisor you are running in your private cloud. Now, they may never get involved in the day-to-day support of the infrastructure work, but they darn-well better understand it because most public clouds are running multiple flavors of virtualization.

**#3 – Broad Technical Background**

Along with virtualization experience, they must understand storage and networking. Why? Because gone are the days when network and storage are silos. You need people who can design a solution that scales and performs with high availability and uptime. Applicants also need to understand fault tolerance and failure domains so they are not putting all the eggs in one basket.

**#4 – Scripting Guru**

Have I said they need to be able to script yet? Bash, Powershell, Perl, Ruby, JavaScript, Python – you name it. They must be able to write code to automated repeatable processes. But we’re not stopping there because they also need to be able to code to RESTFUL APIs. That’s right, if you are going to replace manual processes such as assigning IP addresses and DNS reservation, someone needs to write some code.

**#5 – Borderline Developer (more is better)**

Have I said they need to code in C+, C++, .NET, ASP? No, I am not repeating myself. I am talking about writing scripts that will fire off and orchestrate the complete deployments of DEV, QA and Production environments via tools such as Chef, Puppet, CFEngine or other tools of this kind. Why? Because gone are the days when someone installs Windows or Linux from a CD. Nowadays, you fire off a command that shoots out a server build, then triggers another script that installs applications, then licks its lips and shoots off yet other scripts that do configurations and validation checks. Whom do you think is going to write all this code? Not a SysAdmin. DevOps Engineers will.

I will answer this question by taking two possibilities:

1. You already are a DevOps Engineer but want to polish your skills
2. You are about to enter DevOps culture and want to become a good Devops engineer from the beginning

For the first scenario, try to get hands-on on all the tools one by one and try to adapt new top trending tools and keep updating yourself. This will help you to gain some confidence and help you to retain the interest.

You might like to visit below link to get an idea on top trending DevOps tools of 2018.

[Top 10 DevOps Tools You Must Know In 2018 | Edureka](https://goo.gl/T9b3wC)

Top 10 DevOps Tools For 2018:

* + GIT
  + JENKINS
  + SELENIUM
  + DOCKER
  + PUPPET
  + CHEF
  + ANSIBLE
  + NAGIOS
  + ELK STACK
  + SPLUNK

You might also like the video on DevOps automation tools. Below is the link:

[DevOps Tools | Automation using DevOps Tools | DevOps Training | DevOps Tutorial | Edureka - YouTube](https://goo.gl/bLFwiZ)

Now for the second scenario, you basically want to learn DevOps to start and get exponential growth in your career.

This DevOps training tutorial was very helpful for me during my learning. It helped me gain the insights of **DevOps environment** and enlightened me with the information about various tools involved during different processes in DevOps cycle.

[DevOps Training Videos - YouTube](https://goo.gl/K1UcwG)

You might like the blog on DevOps which explains about DevOps environment and other important aspects regarding DevOps. Below is the link.

[What is DevOps - Facebook's UseCase | DevOps Tools | Edureka](https://goo.gl/QH6Ljm)

https://www.edureka.co/blog/what-is-devops/

If you want to dive into the depth of DevOps and learn about the DevOps tools in detail you can click on the link given below. The link contains videos on

* Git
* Jenkins
* Selenium
* Chef
* Puppet
* Ansible
* Docker

[DevOps Tools Tutorial Videos - YouTube](https://goo.gl/3gV7Kz)

You would want to become a certified learner in DevOps course, and for that you can get DevOps certification training from **Edureka**.

DevOps course from Edureka is good enough for you to become an expert in the domain. The course is precisely designed keeping in mind the current industrial requirements. This course will provide you in depth idea about:

* Continuous Development, Continuous Testing and Continuous Integration using **GIT**, **Jenkins** and **Selenium**
* Continuous Deployment using **Puppet**, **Ansible** and **Docker**
* Finally, you will gain insights by continuously monitoring the software using **Nagios**

Here is a link for DevOps certification.

[DevOps Training | DevOps Certification | DevOps Course | Edureka](https://goo.gl/mBhWXU)

I would say learn new things and practice on the existing ones -

There are many online resources which are available and can be referred to for enhancing your knowledge and skills -

Instead of joining a training institute I will advise for going through the tons of material which is available online for free, I am re-using one of my answer to another question which is also relevant here -

I would advise following, instead of joining a institute

**Go through the online resources which are available**

* [https://www.udacity.com/course/intro-to-devops--ud611](http://www.udacity.com/course/intro-to-devops--ud611) - Good online course with sample exercises.
* <http://www.edureka.co/devops> - Online Training covering high level process and tools. (Needs Registration)
* [https://www.edx.org/course?search\_query=devops](http://www.edx.org/course?search_query=devops) – Has no. of courses from MS and Redhat.
* [https://www.codementor.io/devops/tutorial](http://www.codementor.io/devops/tutorial) - Basic Tutorial on DevOps.
* [https://mva.microsoft.com/training-topics/devops#!lang=1033](http://mva.microsoft.com/training-topics/devops#!lang=1033) – Lists no. of courses related to DevOps and various tools, methods used.
* <http://devops.com/> - A good blog, has lots of contents.
* [https://dzone.com/devops-tutorials-tools-news](http://dzone.com/devops-tutorials-tools-news) - Lots of links and tutorials

**Practice through free labs.**

There are lots of free labs available to test out you knowledge on cloud (AWS, Azure) etc. you can leverage these to get some knowledge around it.

* [Build Your Server Lab in the Cloud](https://blogs.technet.microsoft.com/keithmayer/p/earlyexpertlabsetup_azure/)
* [Home | Qwiklabs](https://qwiklabs.com/)

**Refer to books available**

There are many books available which can be refereed to including mine which you can get from [Buy Getting Started with DevOps Book Online at Low Prices in India](http://www.amazon.in/Getting-Started-DevOps-Subodh-Jain/dp/8192898466/ref=sr_1_1?ie=UTF8&keywords=getting+started+with+devops&qid=1484204494&sr=8-1).

**Get some certification done**

Looks for this space for more information on that - [DevOps - Amazon Web Services (AWS)](https://aws.amazon.com/devops/)

**Get some certifications -**

AWS also has a separate certification into DevOps and One can also do it to get more relevance into the DevOps space using by leveraging AWS.

Read more at - [AWS Certified DevOps Engineer - Professional](https://aws.amazon.com/certification/certified-devops-engineer-professional/)

Azure has its own set of information on Integration with DevOps , Read more at - [Get started with Azure Devops Integrations | Microsoft Azure](https://azure.microsoft.com/en-in/try/devops/)

**Blogs -**

(My Own Blog) - [DevOps360 - A comprehensive blogs on DevOps practices, methods and tools](https://devops360.wordpress.com/)

[The DevOpsGuys Blog - Talking About All Things DevOps | DevOpsGuys](https://www.devopsguys.com/blog/)

[IT Best Practices: The Beginner’s Guide via @BMCSoftware](http://www.bmc.com/blogs/categories/devops/)

**My Linked In Post -**

[https://www.linkedin.com/pulse/t...](https://www.linkedin.com/pulse/top-7-sites-you-can-learn-devops-free-subodh-jain)

and Lastly you can always Google and get more…

We soon be staring our own Training and Certification Program on DevOps

watch this space for more information on that - [www.devops360.in](http://www.devops360.in/)

To become DevOps Engineer you need skills as below.

1. Good hold on Linux.
2. Linux and windows server administration and monitoring skills.
3. Great troubleshooting skills.
4. Good hold on automation tool like Jenkins.
5. Good hold on container orchestration services like docker.
6. Good hold on container management services like Kubernets, DC-OS , Mesos - Marathon .
7. Good at Infrastructure as code (Chef, puppet, anisible, terraform)
8. Good knowledge of service integration with other tools.
9. Good knowledge on CD/CI flow. (Continuous integration and continuous deployment).
10. Good hold on shell script / python script / any programming language.
11. Proper understanding of git.
12. Good at Solutioning, planning.
13. The courage to take responsibility on own shoulders to deliver project.

How Good you are I DevOps: <http://blog.shippable.com/how-to-be-a-great-devops-engineer>