The Must Know Checklist For DevOps & Site Reliability Engineers

Since the word DevOps has made a market trend, I've been receiving several emails and phone calls almost everyday about DevOps engineering open positions.

For evangelists, DevOps is a culture and a transformation. For some engineers, DevOps is a set of agile tools and techniques. For managers, DevOps is a probably a methodology. For other people it is just a buzzword and for recruiters, DevOps is a job.

I think DevOps is not just a buzzword but somehow it is a mix of all the above definitions: There is no digital transformation without the right methodologies, the right tools and the right engineers.

In a previous post I wrote The 15-point DevOps Check List that described an approach to implement DevOps in a business. This post is more about the skills required for a DevOps, Site Reliability Engineers and site engineers. Many of the points listed below are specific to *nix environments, **this doesn't mean that there is now DevOps environment for windows lovers.**

This list is **not exhaustive** but **enumerates only technical basic, must-know skills** and some **random thoughts**. You may use them as a checklist **to evaluate yourself** or **someone else** or to **prepare** for your next **DevOps/SRE job interviews**.

This list is **opinionated.**

- You should master *nix systems and have a good understanding of how Linux distributions work.
- Be at ease with Terminal. You may have GUIs to manage your servers but you have to LOVE the terminal no matter what's the case, it is faster, secure and honestly it is easier once you master it.
- How to get the CPU/system info (cat /proc/version, /proc/cpuinfo, uptime, et. al.)
- How cron jobs works. Set cron jobs on specific days/time/month.
- How to know what OS you are running on your machine (cat /etc/lsb-release)
- Learn the difference between different *nix OSs and How to know what OS you are running on your machine (e.g. cat /etc/lsb-release)
- Difference between shells: sh/dash/bash/ash/zsh ..
- How to set and unset ENV variables. Exporting ENV variable is temporary, how to export permanent variables ?
- What are shell configuration files : ~/.bashrc, .bash_profile, .environment .. How to "source" settings for program initialization files.

- Knowing Vim, its configuration (.vimrc) and some of its basic tips is a must.
- How logging works in *nix systems, what are logging levels and how to work with log management tools (rsyslog, logstash, fluentd, logwatch, awslogs ..)
- How swapping works. What is swappiness. (swapon -s, /proc/sys/vm/swappiness, sysctl vm.swappiness ..)
- How to view/set network configuration on a system
- How to set static/dynamic IP address on a machine with different subnets? (Hint: CIDR)
- How to view/set/backup your router settings?
- How DNS works ? How to set-up a DNS server (Bind, Unbound, PowerDNS, Dnsmasq ..) ? What is the difference between recursive and authoritative DNS ? How to troubleshoot DNS (nslookup, dig ..etc)
- Get familiar with DNS and A, AAAA, C, CNAME, TXT records
- How SSH works, how to debug it and how you can generate ssh keys and do passwordless login to other machines
- How to set-up a web server (Apache, Nginx ..)
- What are the difference between Nginx and Apache ? When to use Nginx ? When to use Apache ? You may use both of them in the same web application, when and how ?
- How to set-up a reverse-proxy (Nginx ..)
- How to set-up a caching server (Squid, Nginx ..)
- How to set-up a load balancer (HAproxy, Nginx ..)
- What is an init system ? Do you know Systemd (used by Ubuntu since 15.04) , Upstart (developed by Ubuntu), SysV ..
- Get familiar with Systemd and how to analyze and manage services using commands like systemctl and journalctl
- Compiling any software from its source (gcc, make and other related stuff)
- How to compress/decompress a file in different formats via terminal (mostly: tar/tar.gz)
- How to set-up firewalls (iptables at least ufw) : set rules,list rules, route traffic, block a protocol/port ..
- Learn most used port numbers on which services runs by default (like: SSH (22), Web (80), HTTP/S (443) etc.)
- Learn how to live debug and trace running application in production servers.
- Be at ease with scripting languages. Bash is a must (Other scripting languages are very useful like Python, Perl..).
- Learn how to use at least one of the configuration management and remote execution tools (Ansible, Puppet, SaltStack, Chef ..etc). Your choice should be based on criterias like: syntax, performance, templating language, push vs pull model, performance, architecture, integration with other tools, scalability, availability ..etc.
- Learn how to configure and use continuous integration and continuous delivery tools like Jenkins, Travis CI, Buildbot, GoCd. Integrating this tools with other tools (like Selenium, build tools, configuration management software, Docker, Cloud providers' SDKs ..etc) is helpful.
- Learn distributed version control system Git and its basic commands (pull/push/commit/clone/branch/merge/logs…etc.). Understand git workflows. Do you know how to revert a Git repository to a previous commit ?

- How to use SSH-keys. Try Github, Bitbucket or Gitlab .. to configure passwordless access to the repo/account
- Get familiar with tools like Vagrant to help you create distributable and portable development environments.
- Start looking into infrastructure as code and infrastructure provisioning automation tools like Terraform and Packer
- Start looking into containers and Docker. It's underlying architecture (cgroups and namespaces). How it works?
- Start getting familiar with basic Docker commands (logs/inspect/top/ps/rm). Also look into docker hub (push/pull image)
- Start looking into container orchestration tools: Docker Swarm, Kubernetes, Mesosphere DC/OS, AWS ECS
- Dive into DB (MySQL or any other which you like)
- Learn about Redis/Memcache and similar tools
- What is your backup strategy ? How do you test if a backup is reliable
- Do you know ext4, ntfs, fat ? Do you know Union filesystems ?
- Develop your Cloud Computing skills. Start by choosing a cloud infrastructure provider: Amazon Web Services, Google Cloud Platform, Digitalocean, Microsoft Azure. Or create your own private cloud using OpenStack.
- What about staging servers ? What is your testing strategy Unit Testing ? End-to-end ? So you really need staging servers ? Google "staging servers must die".
- Read about PaaS/Iaas/Saas/CaaS/FaaS/DaaS and serverless architecture
- Learn how to use and configure Cloud resources from your CLI using Cloud Shells or from your programs using Cloud SDKs
- Are you familiar with the OSI Model and the TCP/IP model specifications ? What are the difference between TCP and UDP ? Do you know vxlan ?
- Master useful commands like process monitoring commands (ps, top, htop, atop ..), system performance commands (nmon, iostat, sar, vmstat ..) and network troubleshooting and analysis (nmap, tcpdump, ping, traceroute, airmon, airodump ..).
- Get to know HTTP status codes (2xx, 3xx, 4xx, 5xx)
- Are you familiar with IDEs (Sublime Text, Atom, Eclipse ..) ?
- Network packet analysis: tcpdump, Wireshark ..
- What happens exactly when you hit google.com in the browser? From your browser's cache, local DNS cache, local network configuration(hosts file), routing, DNS, network, web protocols, caching systems to web servers (Most basic question yet difficult if goes deep).
- Get familiar with the mumble-jumble of Kernel versions and how to patch them.
- Get familiar with how SSL/TLS works and how digital certificates works (https)
- Get familiar with secure protocols: TLS, STARTTLS, SSL, HTTPS, SCP, SSH, SFTP, FTPS ..
- Know the difference between PPTP, OpenVPN, L2TP/IPSec
- How to generate checksums (md5, SHA ..) to validate the integrity of any file
- Get to know the difference between Monolithic and Microservices architecture.
- Get to know the pro/cons of Microservices architecture and start building similar architectures

- Do you know what is ChatOps ? Have you tried working with one of the known frameworks ? Hubot, Lita, Cog ?
- How do you make zero downtime deployment ? What is your strategy to make rollbacks, self-healing, auto-scalability ?
- Get familiar with APIs and services: RESTfull, RESTful-like, API gateways, Lambda functions, serverless computing,SOA, SOAP, JMS, CRUD ..
- Read about stateless and stateful applications
- Read about DevOps glossary (Google it)
- How to secure your infrastructure, network and running applications ?
- Learn how set-up, configure and use some monitoring systems (Nagios, Zabix, Sensu, prometheus..etc)
- Read about Open Source and how you can contribute to Open source projects. For more info: http://jvns.ca/blog/2016/10/26/a-few-questions-about-open-source/
- You should be able to do post-mortem if something bad happens to your systems. Make a detailed documentation about what went wrong and how we can prevent not to let it happen in future again.
- Try to learn the approach how experts from StackOverflow are solving any problem. Always remember, it's the technology which keeps on changing not the basics. Basics always remains the same.
- Make Google, StackOverflow, Quora and other professional forums your friends.
- Try to establish good development practices and a solid architecture.
- Learn how to scale at production level.
- Follow open-source projects (Kubernetes/Docker etc.) or what excites you.
- Ask questions/doubts/issues on mailing-lists/public forums/tech meetups and learn from those.
- Follow like-minded folks from the community and be updated with latest tech trends.
- Follow some decent tech companies engineering blogs (We follow: Google/Uber/Quora/Github/Netflix). This is the place from where you can learn straight from the experts and get a chance to see their approach to solve any problem.
- Browse a few aggregators like Reddit, hackernews, medium .. etc.
- Follow like-minded developers and tech companies on twitter. (I am always reading articles and watching talks/conferences, post-mortems are some of my favorite content. I also follow a few github repos to see what's going on with the technology that I use.)
- Read various technology related blogs and subscribe to DevOps Newsletters.
- If possible attend local area meetups and conferences. You will get a chance to learn a lot from seniors and others.
- Learn about scalability and highly distributed systems. How to keep them UP and Running all the time?
- Last but not the least… Read Books.

If you have the majority of these skills, you can be sure you have the prerequisites to DevOps, SRE and system engineering.

You can’t learn all these in one-shot. But having a mind-set is the main thing. It will surely take time even to get familiar with all these but as they say journey has the fun. You will fail many times, learn from your mistakes and don’t repeat them.

Always remember, we all are learners here. We learn by hit and trial. Don’t feel shy in failing because that’s how we learn.

We will be happy to hear your suggestions to add other points to this list.