7 Daily DevOps Tools

When you're first getting started with DevOps, things can be extremely confusing and overwhelming. There are a plethora of DevOps tools out there, each with their own use-cases and benefits, but knowing which tools are worth your time and which are unnecessary can only be truly determined with hard-won experience.

This list will let you short-circuit that evaluation cycle and get started with the absolute basics that you need to do DevOps in the field. This isn't a list of "the only tools you'll ever need", that could never be a truthful statement. But it is a list of the only tools you need to get started. These are the tools that I use in my day to day work, and I only need to reach for more specialty tooling on rare occasions.

The Tools

1. Git

Homepage: https://git-scm.com/

Git is the most important tool I use with every single day.

I use it when making PRs for infrastructure and configuration updates in application stacks, I use it when reading code for a code review, and I use it for storing my own scripts and infrastructure as code (IaC).

Unless you have a very good reason not to, even your one-off scripts should get stored in Git and pushed to a remote repository for safekeeping. The first time you experience a disk failure that wipes out your local copies of well-used scripts, you'll appreciate having the ability to just clone out new copies.

Naturally, don't commit sensitive credentials to your repository. Once they're in, they're never coming out.

2. jq

Homepage: https://stedolan.github.io/jq/

jq is awk for JSON.

For those who are unfamiliar with awk, jq is a JSON parser and transformer, allowing you to read, fold, spindle, extract, or anything else you might need to do with JSON. JSON has become the standard format for exchanging data that needs to be machine readable, especially to or from APIs, so having jq in your toolbelt will immensely increase your capabilities.

3. Terraform

Homepage: https://www.terraform.io/

Terraform is one of the best IaC solutions I've ever used.

The flexibility it provides for creating resources across multiple vendors is second to none, and the HCL configuration language is extremely straightforward to read. If you can name a cloud vendor, it's almost certain they have a provider already available for it. If not, and they have an API, you can even write one yourself.

4. Terragrunt

Homepage: https://terragrunt.gruntwork.io/

Terragrunt is a wrapper tool around Terraform that enhances it with additional features that improve quality of life for working with multiple modules at once.

Instead of needing to duplicate a bunch of your configuration code across all of your environments, since Terraform is focused on a single module at a time, Terragrunt lets you focus on just the pieces of configuration that need to change between those environments.

Yes, you can use Terraform without it, and I do so, but for more than a one-off setup, Terragrunt is well worth your time to learn and incorporate into your toolkit.

5. Docker

Homepage: https://www.docker.com/

I fought for years against using containers, instead preferring full virtual machines and bare metal hardware.

I was an idiot.

I have since learned my lesson, and thoroughly embraced containers for the speed and flexibility they provide. They aren't perfect for every use case, but even if you never build a container of your own, most tools and services in the open source community have a pre-packaged container on Docker Hub ready to be used immediately, no installation required.

Most of the webdev world has moved to using containers, however, so learning to use them is essential for supporting dev teams.

6. AWS CLI

Homepage: https://docs.aws.amazon.com/cli/index.html

This is less of an absolute tool and more of a placeholder. You will inevitably be doing work in some cloud, somewhere. You should absolutely be comfortable and familiar with their official CLI tools, because that's going to be the most straightforward way for you to do one-off things like information gathering or updating a configuration value.

In my case, that cloud is AWS, and I highly recommend it.

7. AWS Vault

Homepage: https://github.com/99designs/aws-vault

If you're using AWS as your cloud provider, you should be also using AWS Vault.

AWS Vault is a simple program that acts as a secure credential provider, storing your AWS API keys in an encrypted form on your system and automating the process of getting temporary usage credentials from AWS with them.

You can get away without AWS Vault when you're just starting out, but once you start adding multi-factor authentication to your access requirements, you'll appreciate how it automates that process for you as well. The secure storage of your dangerously-powerful API credentials is a huge plus, even if you aren't using MFA just yet.

And that's a wrap!

Go forth and start doing DevOps! This is a field that is best learned by practicing the craft, so take the tools you've been given and start hacking away. You'll learn more building a single project from scratch than you will just reading articles and documentation, and you'll build a lot more confidence as well.

Questions? Drop me a line: joe@solo-devops.com