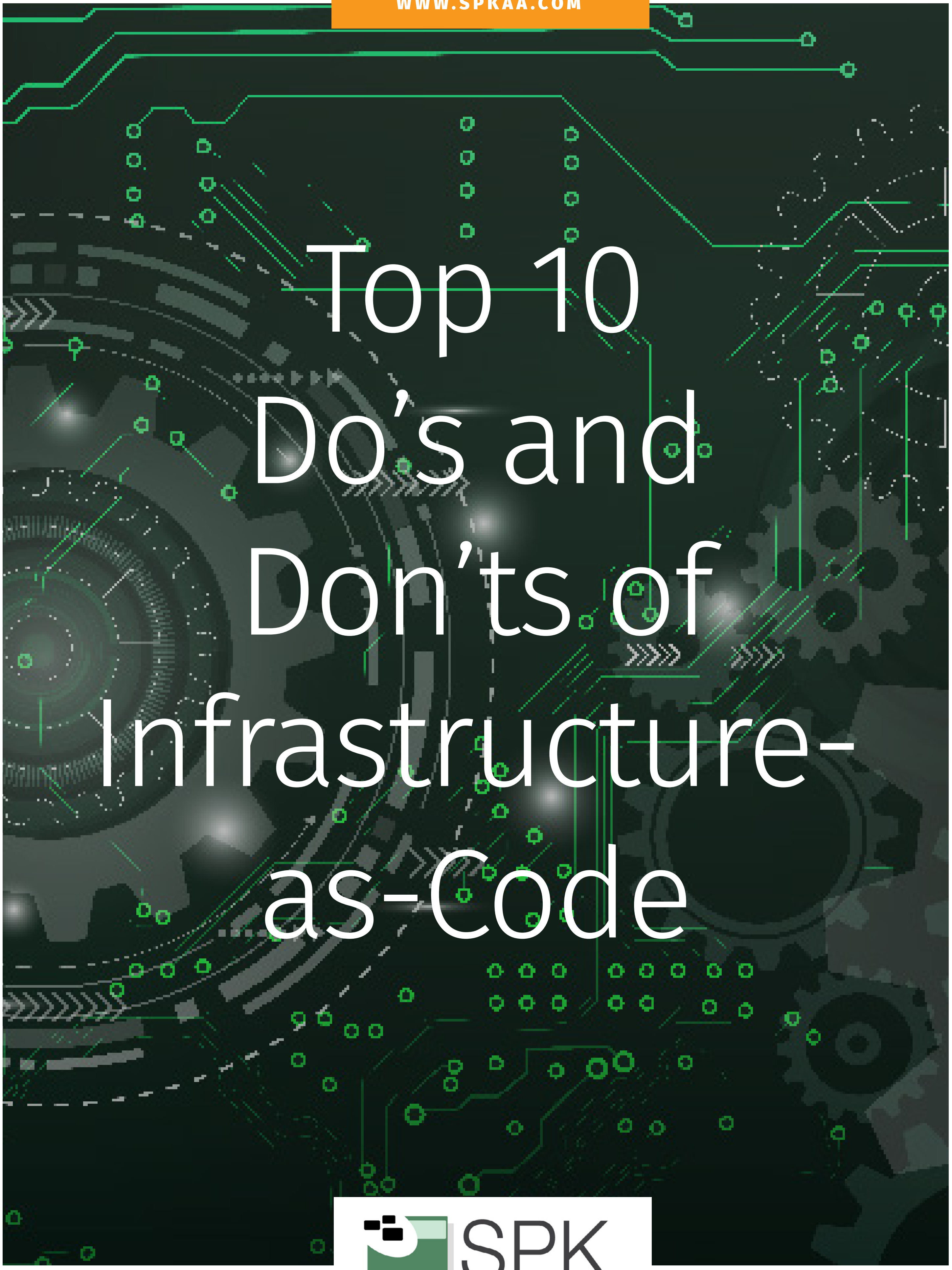
WWW.SPKAA.COM

Top 10 Do's and Don'ts of Infrastructure-as-Code



SPK and Associates

Contents

Introduction

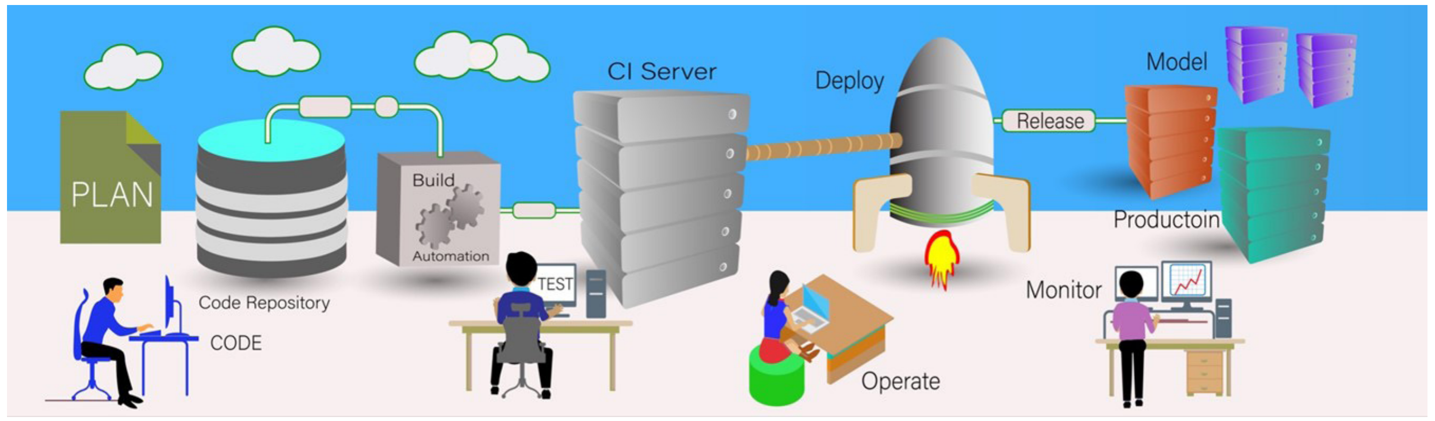
Technology is one of the fastest evolving commodities in human existence. Hardware virtualization began an exciting new adventure in the early 2000s, and cloud computing began its journey to provide the endless scalability options we know today.

In 2006, Amazon Web Services Elastic Compute Cloud was launched which became a pivotal catalyst for questioning the sole methodology of on-premises computers. As cloud computing evolved, so too did the opportunity for businesses and DevOps professionals to consider the future of infrastructure. And with that IaC was born.

What is Infrastructure-as-Code (IaC)

Microsoft describes IaC as:

Infrastructure as Code (IaC) is the management of infrastructure (networks, virtual machines, load balancers, and connection topology) in a descriptive model, using the same versioning as DevOps team uses for source code. Like the principle that the same source code generates the same binary, an IaC model generates the same environment every time it is applied. IaC is a key DevOps practice and is used in conjunction with continuous delivery.



Put simply, *Infrastructure as Code is a descriptive model that resolves traditional infrastructure limitations through programmable code.* IaC tells your Cloud Platform, such as AWS, what your infrastructure looks like and what should be deployed - consistently. It automates the configuration and deployment of infrastructure, without the need to increase CapEx or OpEx. And as IaC is in a pre-configured container, it can be deployed remotely, perfectly version-controlled, and removes the risk of manual "hands-on" configuration, thus removing configuration drift occurrences and human error rates.

What problem does IaC solve?

Think back to 20 years ago when a business, let's call it "Brand A", wanted to create an amazing web application for its customers to visit. The first course of action would be to provision a few on-premises computers to both build and deploy the website. So, Brand A sets off to provision these two computers, configures them with the exact same specifications, and hires a genius web developer to create a website that will attract customers for them. Fast forward five years, and Brand A has grown. It now needs a faster website to deal with the high volume of web traffic. So, Brand A invests in new computers, waits a few weeks for them to arrive, configures the computers and the web application to replicate the original, and then starts the process again.

But this time around, Brand A also needs staging computers, and the IT team now wants computers dedicated to testing environments too. This way the code can be tested before it's put into staging, and then before it is pushed to the live environment. For each live environment computer, you need the same volume of staging computers. For testing computers, you may need a quarter of that.

Now you can see that in order to build, test, stage, deploy and host we encounter the following problems with a traditional infrastructure approach:

Time spent

Not only does Brand A have to wait for hardware to arrive, but they also have to factor in repeated labour time to continually replicate environments. They also need to factor in developers leaving and potentially taking the knowledge with them, along with retraining new developers in their specific environment.

Cost

Hardware and developer time costs money. The other downside is the dprecia-tion of hardware as technology quickly evolves.

Scalability

Growth is determined by your CapEx, OpEx and hardware expedience. Scaling at speed is needed to be a key player in any market.

Error rates

Even the most skilled developer can make mistakes with tedious and repeated configurations over multiple devices. Failure to replicate environments in perfect synergy is referred to as configuration drift or environment drift. When drift happens it can lead to each machine becoming an individual snowflake in the overall system.

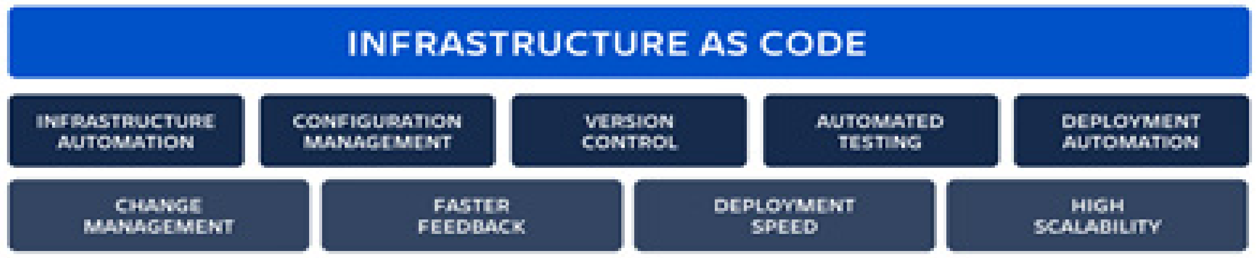
Traceability

When configuration drift occurs, identifying where the version control failed can become a full-time job in itself.

Infrastructure as Code has become pivotal in the way that businesses provision, configure, test and deploy infrastructure, and therefore their products. Cloud tools such as Terraform, Ansible, AWS Cloud Formation and others are becoming dominant infrastructure options for businesses. We are moving from the age of iron, to the age of the cloud.

Businesses still solely leveraging traditional on-premises infrastructure are finding themselves left behind with slower releases and time to market. The competition utilising IaC is now thick and fast.

Atlassian has a simple diagram that helps to describe the way IaC works:



What is the difference between IaC and IaaS?

Whilst they sound similar, Infrastructure as Code, and Infrastructure-as-a-Service are actually two different concepts:

1. Infrastructure-as-Code - A tool used to provision, manage and deploy infrastructure, either in the cloud or on-premises.

2. Infrastructure as a Service - Part of the core cloud services that includes virtualized computing resources like networking, servers, storage, etc.

Think of IaaS as the brain. It's there to act as the container for managing and storing functionality. IaC is the nervous system. It creates the nodes and firing mechanisms to activate the body of your business.

Leveraging both IaC and IaaS can drastically improve the management of your business infrastructure and your product quality.

For SMEs (small and medium enterprise), navigating the new technological developments in order to compete in tomorrow's markets, whilst still meeting your customer and market demands today, can be a distraction you wish to deal with later. And even Enterprises are no exception to this.

However, failing to start down this path to improve your infrastructure and technological advancements **now** can leave you even further behind. So too can "dabbling" in IaaS and IaC without a thorough strategy. Many of our clients who have dedicated in-house teams still prefer to outsource the exploration of IaC and IaaS.

If your business does not have in-house capability or availability to protect your business needs tomorrow, it is highly recommended that you choose a trusted infrastructure and DevOps partner, like SPK and Associates, to support your vision for tomorrow, whilst you focus on your business vision for today.

How does IaC align with DevOps?

Any good modern software and dev team knows that building rapidly, reliably and with continuous learning and evolution is what makes a business grow. And any good software and dev team knows that cherry-picking the processes ripe for automation and removing manual laborious tasks allow them to actually focus on more business growth.

So you can see why at first glance it's easy to jump to the conclusion that IaC is just another DevOps automation tool. But, it's actually so much more than that. Think of the underlying principles of DevOps:

Collaboration

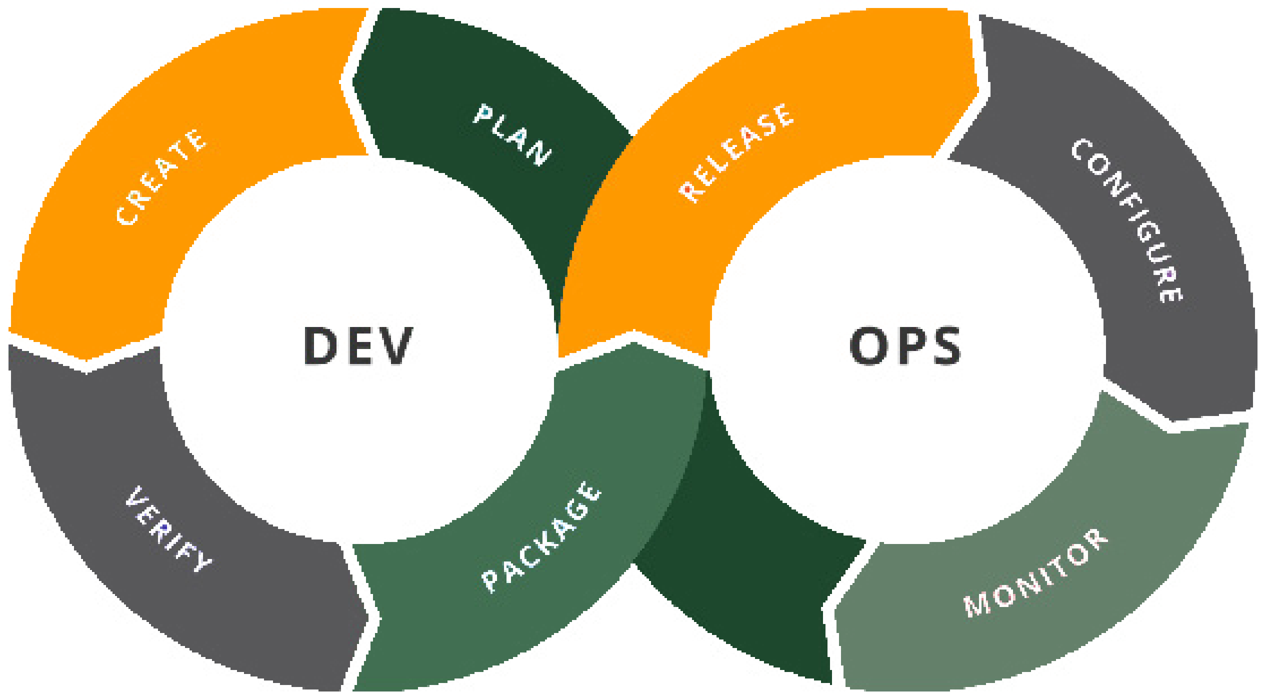
Version control

Virtualized tests

Continuous monitoring

Automation

These can all be applied to your infrastructure in the same way they can be applied to any other code your DevOps team manufactures.



IaC is usually deployed by:

1. DevOps defines and writes the infrastructure specifications;
2. The files are sent to an API, master server, or code repository;
3. An IaC tool such as Terraform creates and configures the specified computing resources.

With the success of continuous improvement/ continuous development (CI/CD), and the ability to release intermittent updates and bug fixes, utilising IaC in conjunction with DevOps practices becomes a holy grail for increasing time to market.

What are the benefits of Infrastructure as Code?

Infrastructure changes are expensive. But in order to remain relevant, and even ahead of the curve, infrastructure becomes a non-negotiable commodity that requires constant evolution to maintain a competitive advantage. Since physical hardware is expensive, organizations try their best to future-proof infrastructure as it is implemented by overbuilding and creating additional investment needs. However, they typically find that IT evolves so quickly that the future-proofing is already out-of-date.

IaC simplifies change management and easily compensates for frequent changes being delivered consistently. This removes the need to over-future-proof system requirements.

Through IaC, your business can benefit from:

- ✓ **Consistency** - Environments are replicated perfectly and with accurate script version controls.
- ✓ **Risk mitigation** - As the environments are within pre-defined containers, ready for deployment and require a hands-off approach to replication, human error is replaced by seamless automation.
- ✓ **Cost optimization** - Improved profit margins due to lower CapEx and less labour intensity for change management.
- ✓ **Agility** - With the ease of traceable flexibility, pre-defined hosting, and ready-to-deploy tools available globally, IaC can improve accuracy and speed to market.
- ✓ **Elasticity** - IaC provides unimaginable scalability potential for businesses without the need to increase CapEx or OpEx.

What are the challenges of IaC?

So we are now on the journey of understanding exactly why your business should consider IaC, including its ease of reproducible and traceable environments.

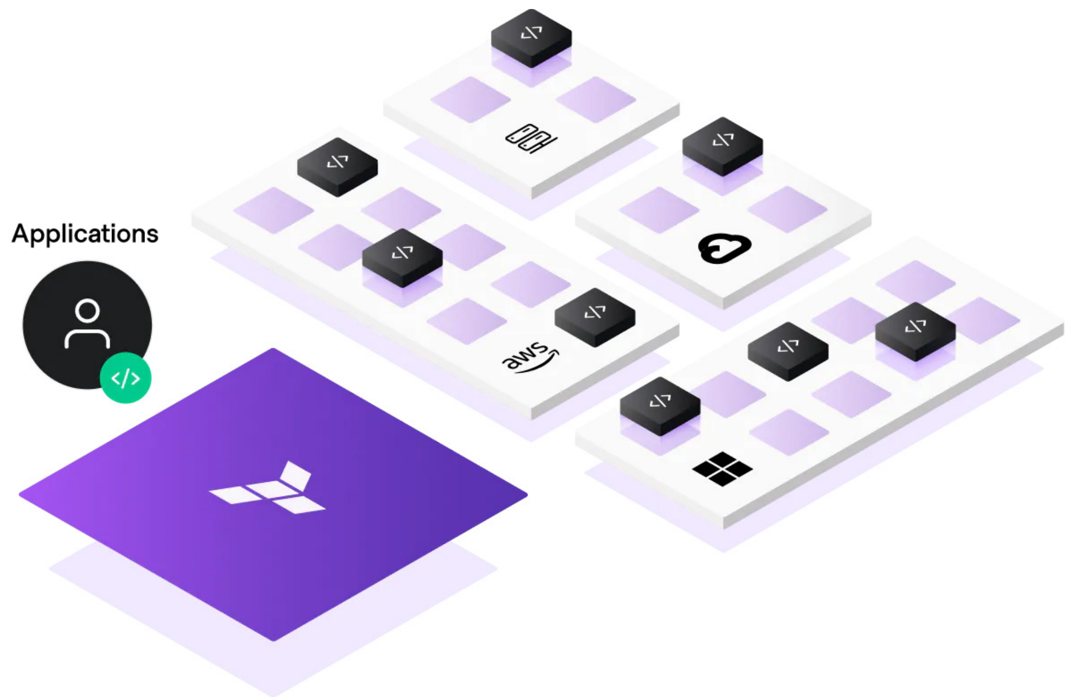
But, as with any technological evolution, there are always challenges to consider too. Whilst Infrastructure as Code has changed the game in hardware virtualization, before transitioning to this digital transformation it is also important to consider the following:

- **Increased complexity** - Whether your IaC scripts are written in HashiCorp Configuration Language (HCL) or plain Python or Ruby, your DevOps team still needs to be fluent in this language and declarative approach for conventions to be accurately applied and traced.

- **Traceability vs. scalability** - Whilst IaC opens up a world of scalability and improved version control, it too becomes a complex beast when hundreds of developers are involved.

- **Feature Lag** - Regardless of the IaC tool, as new cloud features are released, you may be privy to tooling that lacks the access to new cloud features, unless you extend the functionality yourself.

Choosing an IT partner with expertise to manage these on your behalf is ultimately the go-to for many established businesses and larger enterprises.



Terraform IaC tool - Use Case

At SPK, we use Terraform as our IaC tool. Terraform is an open-source IaC tool created by HashiCorp and provides a declarative approach to infrastructure. Terraform is described as:

Terraform allows infrastructure to be expressed as code in a simple, human readable language called HCL (HashiCorp Configuration Language). It reads configuration files and provides an execution plan of changes, which can be reviewed for safety and then applied and provisioned.

Extensible providers allow Terraform to manage a broad range of resources, including IaaS, PaaS, SaaS, and hardware services.

The problem

Our client provides financial services to businesses.

Their primary focus was to develop their financial startup application and ensure that enough features were incorporated for potential customers to be interested. The client's CEO brought SPK on board for our DevOps technical ability, our proven reputation for rapidly growing environments within regulated industries, and our ability to handle the application operations to ensure that robust, stable, proactive monitoring and alerting was in place.

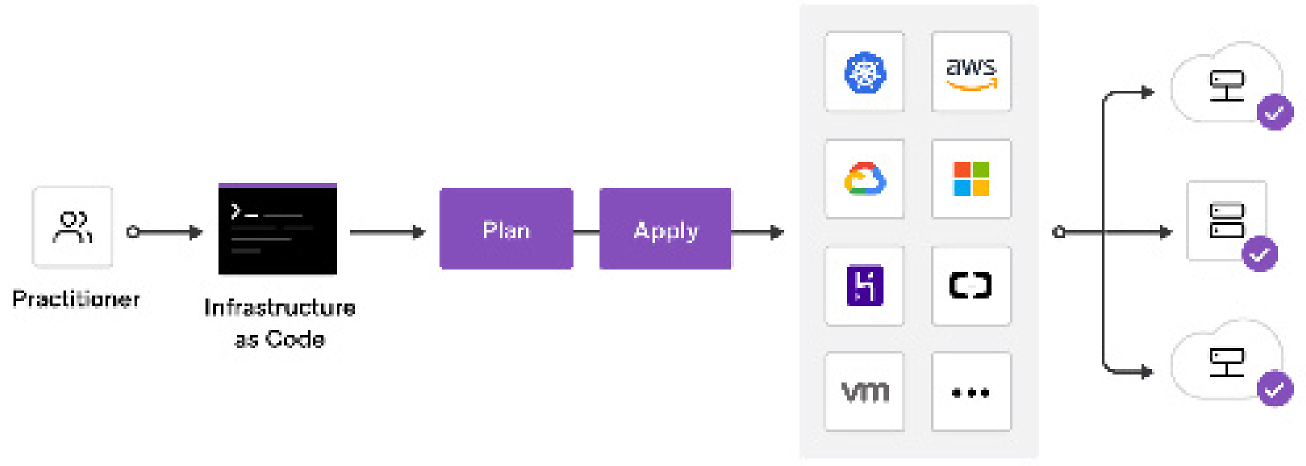
Additionally, it was critical that the application is easily replicated – they would need demo environments, development environments, test environments, etc. If there were any inconsistencies between environments, the application might behave unexpectedly.

Had they not implemented a solution, many hours would have been expended configuring multiple environments manually. This would have led to the risk of introducing human error. Furthermore, without having implemented all of the operational best practices suggested by SPK, their application platform would be much less stable, and outages would occur as a result.

Identifying the path to success

The client had expressed their primary business objectives. SPK then translated these into technical requirements. We presented two separate options along with the pros and cons of each, and with the client's agreement, we moved forward with the solution.

The option our client chose to proceed with included Terraform, Terraform Cloud, Atlassian Bitbucket and Bitbucket Pipelines, ELK Stack, and Grafana. This option would require two engineers - one of which also played a Primary Customer Program Manager role.



The solution

SPK began to develop Terraform code based on the client's current production environment which was created manually. We inventoried all AWS resources and redefined them within Terraform. These resources included Elastic Beanstalk environments, email services, roles, users, DNS records, and SSL certificates. Once the Terraform code was established, we could create new, complex environments within minutes.

Shortly thereafter, SPK had a robust operational environment that consisted of Terraform automation, proactive monitoring and alerting.

Two SPK staff on a fixed monthly fee saved the company roughly $85K per year, compared to two full-time staff plus benefits and company equity.

Top 10 Do's and Don'ts of Infrastructure as Code

1. Use an iterative approach.

Chances are that DevOps automation teams will be introduced into situations where the applications they are supporting have already been developed to a degree. These teams do not get the luxury of being able to develop their IaC from scratch in lock step with the application development.

Therefore, it may seem daunting trying to decide exactly where to start. Networking configuration is a good candidate. It forms the platform basis, and may not change much, if at all. Incorporating incremental changes over time allows for thorough testing cycles as the IaC automation gradually impacts more of the environment.

2. Be vigilant regarding credential storage

When writing application code, developers go to great lengths to ensure passwords and credentials aren't stored alongside the application code, in the source code repository. Additionally, ensuring that application files have the correct permissions or are only accessible by the appropriate staff guarantees that backend systems aren't being accessed inappropriately.

IaC should be treated the same way. Leverage services such as AWS Secrets Manager to fetch credentials dynamically at run-time. Or, include SCM post-commit hooks to ensure that code is scrubbed before check-ins.

3. Leverage cloud-enabled collaboration tools

In the case of Terraform, the state of the systems under control is represented by a state file. This state file can pose a challenge when a team of individuals are all attempting to work in the same environment. Terraform Cloud can help alleviate this by maintaining the state file in a shared repository. Additionally, each team member can see operations in progress by other team members. Finally, Terraform Cloud offers great integration into SCM tools whereby Terraform changes can be applied as soon as a new code change is applied to the repository.

4. Don't attempt to automate everything

It can be quite satisfying to see entire, complex environments being represented by a set of source code files. However, some things aren't well suited to being controlled by IaC – and it can be a gray area. For example, should application databases and schemas be created by the IaC tool, or should they be managed by the application itself?

Perhaps the delineation is that the application will always expect a database to be present – so that should be created by the IaC tool. But the database schema is far too dynamic, and is better suited to being managed by the application. In general, it helps to consider what ultimately will lead to fewer blockers, and what will help accelerate application development.

5. Foster a self-service approach with the application developers

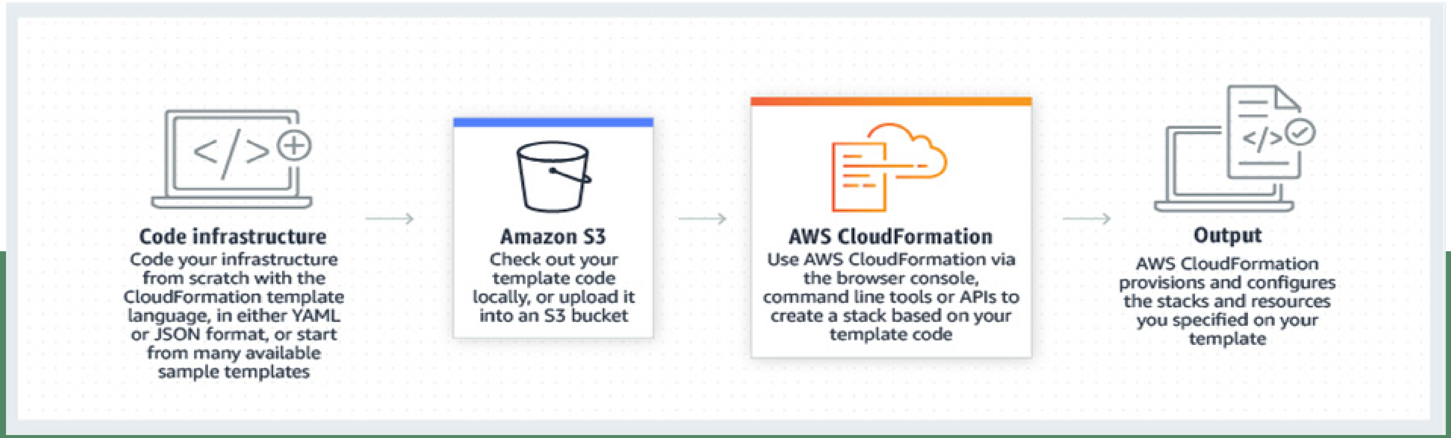
DevOps is truly a marriage between Dev and Ops teams. With that comes shared responsibility, and sometimes these responsibilities can shift between the teams. For simple changes to the environment, the application developers may submit a request to the Ops team. But what if they could make these changes themselves? If platform code and application code are unified, then changing something like a VM instance type would be a quick one line update and code commit. No need for application developers to be blocked due to a minor or low-risk change.

6. Don't forget to audit cloud spend

With increased automation comes the possibility that cloud costs may spiral out of control. Consider implementing guardrails such as resource limits, or take on a trust-but-verify approach by setting up budget alarms or monthly spend reports.

7. Consider platform agnostic tools

IaC tools such as CloudFormation have a very deep integration into their own native cloud platform. This allows for very fine grained control over the environment and requires less involvement of 3rd party tools. This is not a problem until you need to automate a resource that is outside of AWS. Consider platform agnostic tools that can provide good control into a wide variety of platforms: AWS, Azure, VMware, Docker/Kubernetes.


Code infrastructure
Code your infrastructure from scratch with the CloudFormation template language, in either YAML or JSON format, or start from many available sample templates
Amazon S3
Check out your template code locally, or upload it into an S3 bucket
AWS CloudFormation
Use AWS CloudFormation via the browser console, command line tools or APIs to create a stack based on your template code
Output
AWS CloudFormation provisions and configures the stacks and resources you specified on your template

8. Budget for commercial support for mission-critical environments

Getting started with open source IaC tools is great, especially for startups who have yet to reach production and who have limited budgets. However, as the codebase matures, and an increasing number of users begin relying upon your application, consider commercial support from the IaC tool provider. The specialized support can provide quick ROI during a production outage where revenue is lost by the second.

9. Extend functionality using external scripts

There are many downsides in relegating to external bash or python scripts when your IaC tool cannot automate exactly what you're attempting to do – you then have disparate code which is more difficult to follow. Ensuring the script is fully integrated and behaves in a way that is expected by the IaC tool can be a challenge (risk of stability). Another downside is that you now have another set of utilities or software libraries to maintain and keep track of.

However, the benefits can outweigh these downsides. Complex tasks can be easily integrated. Bash and Python are almost universal and can run on almost any system. And these separate pieces of code are still stored alongside the IaC code in your SCM, so there's full visibility and change control.

10. Integrate metrics for BI reporting

Once systems are automated, it is difficult to quantify the amount of time that was saved, or what the overall benefit to the team was. Consider incorporating metrics gathering early in the process. That way a baseline can be established and any improvements to platform deployment time or provisioning time can be recorded over a longer period.

Conclusion

IaC focuses on automating cloud IT infrastructure, but it is not solely about automation. It provides a tool that seamlessly aligns to all DevOps principles and enables business growth elasticity without the need to increase CapEx or OpEx. By implementingInfrastructure-as-Code, businesses can configure, test and deploy with minimal risk of errors and bugs due to configuration drift. It empowers businesses to do more, for less cost.

IaC tools such as Terraform provide ease in managing your infrastructure. However, many clients prefer to partner with external IT expertise to deliver IaaS and IaC. Such a partnership allows them to quickly adapt to the digital transformation required whilst still allowing them to compete in today's market.

SPK and Associates has 20 years of expertise supporting clients in cloud computing and deliveringInfrastructure-as-Code globally across multiple industries and businesses of all sizes.

Is it time for your business to scale-up operations without scaling up your CapEx and OpEx?

SPK and Associates is focused on improving Engineering with smart information technology solutions. SPK understands the systems, processes, data and applications critical to successful product development, and dedicate ourselves to helping clients build, test, and release products faster and better. For 25 years, we have helped our customers harness technology to optimize engineering and accelerate product delivery.

spkaa.com

(888)-310-4540

info@spkaa.com

5011 Scotts Valley Drive
Scotts Valley, CA 95066

SPK
and Associates