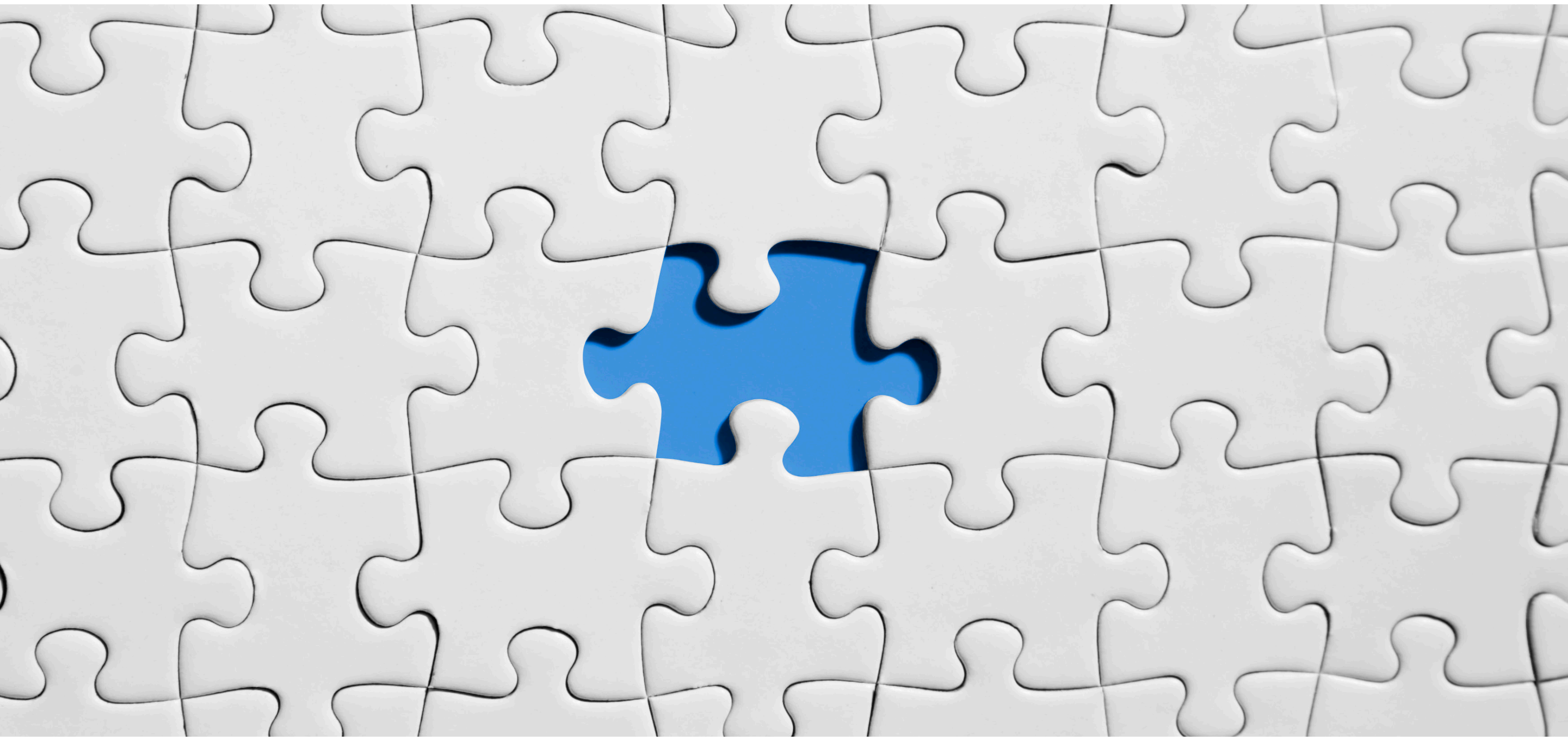
INFRASTRUCTURE AS CODE



Rahul Joseph

Sales Engineer Analyst

Dell Technologies

Rahul.joseph@dell.com

Rani Priya

Sales Engineer Analyst

Dell Technologies

RaniPriya.priyaS@dell.com

The Dell Technologies Proven Professional Certification program validates a wide range of skills and competencies across multiple technologies and products.

From Associate, entry-level courses to Expert-level, experience-based exams, all professionals in or looking to begin a career in IT benefit from industry-leading training and certification paths from one of the world’s most trusted technology partners.

Proven Professional certifications include:

- Cloud
- Converged/Hyperconverged Infrastructure
- Data Protection
- Data Science
- Networking
- Security
- Servers
- Storage
- Enterprise Architect

Courses are offered to meet different learning styles and schedules, including self-paced On Demand, remote-based Virtual Instructor-Led and in-person Classrooms.

Whether you are an experienced IT professional or just getting started, Dell Technologies Proven Professional certifications are designed to clearly signal proficiency to colleagues and employers.

Learn more at www.dell.com/certification

Table of Contents

Introduction

Managing IT infrastructure was and, in some cases, still is a very manual process. Engineers would physically put servers in place and configure them. Only after the machines were configured to the correct setting required by the OS and applications would those engineers be able to deploy the application. This would take days to weeks depending on the technology being implemented. Naturally, these manual tasks would often result in several difficulties.

The first big problem is operating expenditure. You'd need to hire many professionals to perform the required steps at each stage of the process, from network engineers to hardware maintenance technicians to storage admins; the list is never ending. All those professionals must be compensated, obviously, but they also need to be managed. That results in more management overhead and adds complexity to communicate and collaborate within the organization. This leads to huge losses for these organizations. Not to mention the building costs and costs to maintain your own data centers, which would increase the expenditure by orders of magnitude.

The next big problems are scalability and availability. In the ever-changing scenario we currently live in organizations need to be ready to come up with new requirements quickly. Since manual configuration is very time consuming, applications would very often struggle with spikes in access, while the system admins would be desperately trying to line up additional servers to manage the load. This directly impacts availability for users. If the organization didn't have backup servers or data centers, the application might be unavailable for very long periods of time.

Finally, on our list of problems comes inconsistency. We need to make provisioning of infrastructure resources reliable, robust and not prone to human error. Manually deploying configurations often lead to mistakes and if there are several people involved it could lead to discrepancies as well.

Cloud Computing and Infrastructure as Code

The move towards cloud computing – whether private or public clouds – helps solve some of the problems mentioned above. It reduces costs by freeing you from having to build and maintain your data centers and the high cost associated with it. It also provides rapid scaling on demand to support fluctuating workloads.

Although cloud computing allows you to set up your infrastructure quickly and thus solves the problem of scalability and availability, it does not solve inconsistency issues. When there are several people performing the configurations, there's bound to be discrepancies.

To address these challenges, Development and Operations (DevOps) is encouraging continuous collaboration between developers and operations staff by means of a set of principles, practices and tools to optimize the software delivery time [1]. DevOps implies a significant transformation in IT culture, focusing on rapid IT service delivery through the adoption of agile methodologies and lean practices in the context of a system-oriented approach [2].

Infrastructure as Code (IaC) is a method to obtain infrastructure automation built on practices from software development. It emphasizes consistent, repeatable routines for provisioning and modifying systems and their configuration. Modifications are made to definitions, and then rolled out to systems through unattended processes that include exhaustive validation [3]. IaC helps manage IT infrastructure using configuration files. Essentially, it enables IT professionals to program infrastructure just as with software.

According to Brikman [4], the use of cloud computing along with the IaC approach is leading to changes, such as:

- Many companies are migrating to the cloud, rather than managing data centers.
- Many IT teams are spending all their time working on software, rather than investing heavily in hardware.
- Many system admins are writing code, rather than racking servers and plugging in network cables.

Although this is true, the shift in provisioning is not only for the cloud. Regardless of the type of infrastructure, IaC enables IT staff to take a programmatic approach to provisioning and eases logging, auditing, and versioning. This provides organizations deeper insight to the precise, current state of their infrastructure, eliminates reliance on error-prone manual practices and creates a single streamlined workflow for provisioning both cloud-based and private data center infrastructure [5].

Benefits of Infrastructure as Code

Here are some key benefits organizations will reap by adopting an IaC solution:

1. **On Demand Self Service**
   Most development teams have a small number of system admins associated with them and they are the only ones with access to production. This becomes a major bottleneck as the company grows. IaC helps automate the entire deployment process enabling developers to provision their own infrastructure as and when they require.

2. **Speed**
   IaC enables quick set up the entire infrastructure by running a script. This can be done for every environment, from development to production, passing through staging, QA, and more. IaC can make the entire software development lifecycle more efficient.

3. **Consistency**
   Humans are fallible and manual processes often result in errors. IaC provides an automated process which is more consistent, more repeatable and not prone to manual error. The config files themselves are the single source of truth, to guarantee that the same configurations will be deployed repeatedly, without discrepancies.

4. **Accountability/Documentation**
   The ability to version IaC configuration files like any source code file provides full traceability of the changes each configuration suffered. IaC acts as documentation, allowing anyone in the organization to understand how things work if the system admin is unavailable.

5. **Version Control**
   It is considered a best practice to Storing IaC files in version control is a best practice that ensures the entire infrastructure history is captured in the commit log. This is a powerful tool for

debugging issues. When an issue arises, first check the commit log, find out what changed in the infrastructure and then resolve the problem by simply reverting back to the previous known good IaC configuration version.

6. **Validation**
   Since the state of an infrastructure is defined in code, a code review can be performed for every single change, multiple automated tests run, followed by passing the code through static analysis tools, significantly reducing the chance of issues.

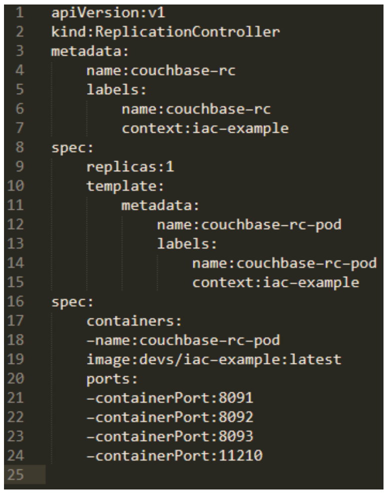
7. **Reuse**
   infrastructure can be packaged into reusable modules [6] to enable building on top of documented, validated configurations rather than doing every deployment for each product in each environment from scratch.

8. **Lower Cost**
   A major benefit of IaC is, without a doubt, lowering the costs of infrastructure management. Employing the latest modern technologies along with IaC dramatically reduces costs, i.e. no need to spend money on hiring people to operate, deploy and manage the infrastructure. IaC also lowers costs in another, subtler way, by ensuring that the smart, high-paid professionals are given more time to innovate and focus on what matters most rather than performing tasks that could be automated.

How Infrastructure as Code works

As mentioned earlier, IaC is the practice of describing complex and (usually) cloud-based deployments by means of machine-readable code. The main enabler for IaC has been the advent of Cloud computing, or more specifically, virtualization technologies which allowed provisioning, configuration and management of computational resources to be performed programmatically [7].

Subsequently, several diverse languages and corresponding platforms have been developed, each of which deals with a specific aspect of infrastructure management. From tools able to provision and orchestrate virtual machines (HashiCorp Terraform, Cloudify, etc.), to those doing a similar job with respect to container technologies (Kubernetes, Docker Swarm), to machine image management tools (Packer by HashiCorp), to configuration management tools (Ansible, Chef, Puppet, etc.). For example, Figure 1 shows a piece of Kubernetes code, which provisions and deploys a noSQL Cloud Database called Cloudbase [7]. This shows how to configure numerous aspects of infrastructure such as the ports to be opened on the host container, the container image which is to be used and the desired number of database clones.



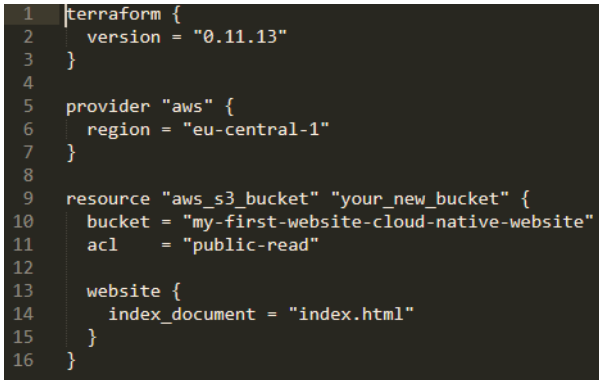
```
apiVersion:v1
kind:ReplicationController
metadata:
    name:couchbase-rc
    labels:
        name:couchbase-rc
        context:iac-example
spec:
    replicas:1
    template:
        metadata:
            name:couchbase-rc-pod
            labels:
                name:couchbase-rc-pod
                context:iac-example
spec:
    containers:
    -name:couchbase-rc-pod
    image:devs/iac-example:latest
    ports:
    -containerPort:8091
    -containerPort:8092
    -containerPort:8093
    -containerPort:11210
```

Figure 1: An example of Kubernetes code

These platforms and tools could also be divided based on the two major modelling approaches it takes:

- Declarative – The declarative approach uses structural models that describe the desired application structure and state, which are interpreted by a deployment engine to enforce this state. That is, you specify the desired final state of the infrastructure to be provisioned and the IaC software takes care of the rest, spinning up the container or virtual machine, installing and configuring the required software, determining system and software interdependencies, and handling versioning.
- Imperative – The imperative approach utilizes procedural models that explicitly specify steps that must be executed, their execution order, and even the data flow between these steps[7]. Such imperative models are executed in an automated way by a process engine. Using the imperative methodology, organizations can modify arbitrarily the deployment but typically requires a skilled administrator to set up and manage, i.e. if multiple different cloud provider application programming interfaces (APIs) must be invoked [7][8].

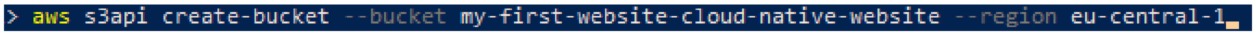
Below are two different ways to provision the same infrastructure. We will create an S3 bucket, first with Terraform (Figure 2), a declarative IaC tool. And second, with ad hoc scripts (Figure 3).



```
terraform {
  version = "0.11.13"
}

provider "aws" {
  region = "eu-central-1"
}

resource "aws_s3_bucket" "your_new_bucket" {
  bucket = "my-first-website-cloud-native-website"
  acl    = "public-read"

  website {
    index_document = "index.html"
  }
}
```

Figure 2: An example of declarative approach in Terraform

When we use the declarative methodology using Terraform we declare the target end state of the required AWS S3 bucket. We don't know if it exists already or not. If it doesn't, the IaC engine will provision one and map it to the requested user. If it already does, no further action will be taken.



```
> aws s3api create-bucket --bucket my-first-website-cloud-native-website --region eu-central-1
```

Figure 3: An example of imperative approach using AWS CLI

On the other hand, the imperative example using AWS CLI has no knowledge on whether this exists already. Hence, a new S3 bucket would be created even if an identical one exists. It also cannot be easily repeated and doesn't have the ability to update or delete the existing infrastructure components.

Commonly used Infrastructure as Code Tools

Some IaC providers are platform-specific, such as Azure Resource manager or AWS CloudFormation. Others are more modular and could be deployed to various platforms, such as Hashicorp Terraform and Ansible. Here we will look in brief at some commonly used IaC tools:

- Terraform – This is a server provisioning tool which can be used to not only create servers, but even databases, load balancers, caches, subnet configurations, firewall settings and almost every other element in the datacenter infrastructure. It is a declarative coding tool that allows developers to use a high-level configuration language called HashiCorp Configuration Language to define the desired "end-state" cloud or on-premises infrastructure for running an application. It then generates a procedure for reaching that end-state and executes the procedure to provision the infrastructure. It is the most adaptable tool in this list and hence quite intimidating to start with.

- Ansible – A configuration management tool designed with automation in mind, Ansible concentrates on providing a "radically simple" configuration language as well as being able to manage cloud instances instantly with no alterations. It also works great for executing arbitrary IT orchestration (e.g. hotfixes, zero downtime rolling updates, etc.) contrary to being configuration management-specific. Rather than managing systems as separate elements, you simply define how elements and the system in general interact with each other and Ansible will take care of the rest.
  Ansible also has a wider support base with Dell Technologies providing Ansible storage integrations to PowerStore, PowerMax and Isilon through Ansible playbooks and plugins [9].

- vRealize Orchestrator and Automation – VMware vRealize Orchestrator (vRO) is an IT process automation tool that allows automated management and operational tasks across both VMware and third-party applications. Dell EMC Storage Workflows for vRO augment the capabilities of VMware's vRealize Orchestrator solution by providing access to storage array-specific management workflows, such as storage administration, provisioning and consumption [9]. vRealize Automation can deploy to an increasing variety of cloud endpoints (AWS, Azure, GCP, vSphere and VMware Cloud on AWS), and blueprints created in vRealize Automation are written declaratively in YAML [11]. Blueprints describe a desired end-state of containers, virtual machines, load balancers, networking and other infrastructure components that are satisfied by the various Cloud Accounts and Integrations.

- Puppet – Puppet is a popular DevOps automation tool for managing the state of infrastructure. Users could use sequential task execution as well as declarative infrastructure state control. Puppet procedures specify the sequence of tasks that need to be executed. The procedures can be defined as YAML files or in Puppet programming language. The Puppet modules below can be used both in Bolt as well as Puppet Enterprise. Puppet Forge is a quickly growing repository of third-party modules written for Puppet execution [9].

- Container Storage Interface (CSI) – Container technologies tremendously increase speed, efficiency and portability of application deployment. However, stateful applications need persistent storage that has a lifecycle independent of the pod that utilizes it. CSI is a standardized API for container orchestration platforms (COs) to communicate with storage plugins for persistent storage. Dell Technologies is building CSI plugins for a growing list of storage platforms to enable new and legacy applications to be deployed in a containerized environment [9].

  AWS CloudFormation – This is a platform-specific IaC tool designed for users working with the AWS Cloud. CloudFormation allows users to model their infrastructure within a JSON or YAML template file. The service also includes automation features to help with deployment of resources in a repeatable and predictable way. Here you only pay for the resources utilized and not for the service itself. With the template configured to your required application specifications, CloudFormation will handle the rest of the tasks for you.

Conclusion

Currently, the landscape of IaC platforms is jeopardized by technology heterogeneity and the large number of available solutions. Although this is the direct result of the immense interest IaC has raised, it complicates the understanding and thereby the adoption of this relatively new technology.

As IaC matures we anticipate it being much more widely utilized and integrated with many more products and offerings from multiple vendors.

References

[1] J. Humble and D. Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, 1st ed. Addison-Wesley, 2010.

[2] DevOps—Gartner IT Glossary URL: https://www.gartner.com/it-glossary/devops/

[3] K. Morris, Infrastructure as Code: Managing Servers in the Cloud, 1st ed. O'Reilly Media, 2016.

[4] Y. Brikman, Terraform: Up and Running, 1st ed. O'Reilly Media, 2017.

[5] HashiCorp Blog: Using Infrastructure as Code to Automate VMware Deployments, URL: https://www.hashicorp.com/blog/using-infrastructure-as-code-to-automate-vmware-deployments

[6] Gruntwork Infrastructure Packages, URL: https://blog.gruntwork.io/gruntwork-infrastructure-packages-7434dc77d0b1

[7] OASIS, Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0, Organization for the Advancement of Structured Information Standards (OASIS), 2013.

[8] Declarative vs. Imperative: Two Modeling Patterns for the Automated Deployment of Applications, Christian Endres, Uwe Breitenbücher, Michael Falkenthal, Oliver Kopp, Frank Leymann, and Johannes Wettinger, URL: https://www.iaas.uni-stuttgart.de/publications/INPROC-2017-12-Declarative-vs-Imperative-Modeling-Patterns.pdf

[9] Dell EMC Storage Automation and Developer Resources, URL: https://www.delltechnologies.com/en-us/storage/storage-automation-and-developer-resources/index.htm

[10] Julio Sandobalin, Emilio Insfran, Silvia Abrah, On the Effectiveness of Tools to Support Infrastructure as Code: Model-Driven Versus Code-Centric: https://ieeexplore.ieee.org/ielx7/6287639/8948470/08959180.pdf

[11] https://www.vmware.com/topics/glossary/content/infrastructure-as-code

[12] https://www.ibexlabs.com/top-7-infrastructure-as-code-tools/