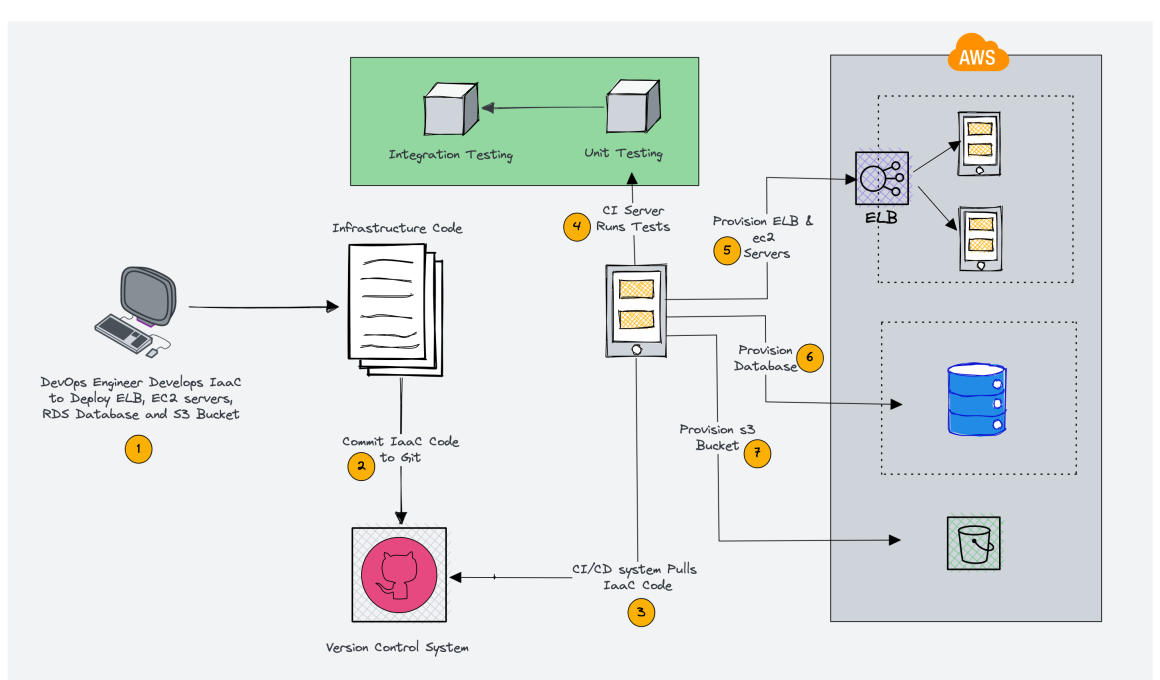
Infrastructure as Code Vs Configuration Management Vs Infrastructure Provisioning

[](https://devopscube.com/wp-content/uploads/2022/12/IaaC.png)

In this blog, you will learn the concept and differences between Infrastructure as Code, Configuration Management, and Infrastructure Provisioning. We will also look at popular tools for each and how they work together.

**TABLE OF CONTENTS**

1. [What is Infrastructure as Code (IAC)?](https://devopscube.com/infrastructure-as-code-configuration-management/#what-is-infrastructure-as-code-iac)
   1. [🚀 Benefits of IaC](https://devopscube.com/infrastructure-as-code-configuration-management/#%f0%9f%9a%80-benefits-of-iac)
   2. [🛠️ IaC Tools](https://devopscube.com/infrastructure-as-code-configuration-management/#%f0%9f%9b%a0%ef%b8%8f-iac-tools)
   3. [🧘 Idempotency](https://devopscube.com/infrastructure-as-code-configuration-management/#%f0%9f%a7%98-idempotency)
   4. [🏋️ IaC High-Level WorkFlow](https://devopscube.com/infrastructure-as-code-configuration-management/#%f0%9f%8f%8b%ef%b8%8f-iac-high-level-workflow)
2. [What is Infrastructure Provisioning?](https://devopscube.com/infrastructure-as-code-configuration-management/#what-is-infrastructure-provisioning)
3. [What is Configuration Management?](https://devopscube.com/infrastructure-as-code-configuration-management/#what-is-configuration-management)
4. [Infrastructure Provisioning Vs Configuration Management](https://devopscube.com/infrastructure-as-code-configuration-management/#infrastructure-provisioning-vs-configuration-management)
5. [IaC & Configuration Management FAQs](https://devopscube.com/infrastructure-as-code-configuration-management/#iac-configuration-management-faqs)

If you are trying to become a [DevOps engineer](https://devopscube.com/become-devops-engineer/), you might get confused between the use cases of **Terraform** and configurations management tools like **Ansible, Chef, Puppet**, etc. Therefore, it is essential to understand the difference between infrastructure as code, infrastructure provisioning, and configuration management.

Let’s look at some of the fundamentals from an infrastructure standpoint before moving on to the key concepts.

1. **Infrastructure**: Infrastructure refers to the combination of hardware and software components that make up the IT environment, such as servers, storage, network devices, firewall devices, routers, and more.
2. **Provisioning:** In IT, provisioning is the process of creating infrastructure and making it available to end users.
3. **Configuration**: It is the process of configuring the provisioned IT infrastructure resources. For example, installing and configuring a database on a server or configuring network and firewall settings.
4. **Automation**: It is the process of automating infrastructure tasks, such as automating the installation of software packages, setting up users and permissions, or configuring network devices.
5. **Orchestration:**Orchestration is the process of coordinating multiple automation tasks. The output from one automation task can be used as input for another, allowing for a more complex and streamlined workflow. For example, to deploy a server, you may first need to provision network resources, and then use the output from that automation (network details) to deploy the server.

**What is Infrastructure as Code (IAC)?**

When it comes it **infrastructure automation**, you often hear the term infrastructure as code. In short IAC.

Traditionally, IT Infrastructure (Servers, storage, network, etc) was provisioned manually or using tools. There was no self-service portal. As a result, a server or network provisioning request might task days to weeks to get fulfilled

But with the advent of cloud computing, provisioning infrastructure has become easy as most of the complex configurations are abstracted away by the cloud providers using virtualization and software-defined networking (Private and public clouds). You can provision network, servers, and storage in a few minutes. And the best part is everything is **API driven**. All cloud providers expose APIs to interact with their platform to provision infrastructure. If it is API driven, you can **use any programing language** to manage your IT infrastructure. Not only just provisioning, but you can also configure the provisioned resources using code.

If you use code to provision and configure the infrastructure, it is called **Infrastructure as code (IaC).**To put it simply, codifying the infrastructure provisioning and configuration. With the concept of Infrastructure as code, you can follow the same workflow you use for application development for Infrastructure as code development. Meaning, versioning the infrastructure code in git, running unit tests, and integration tests, and then deploying it. Over time, many tools have evolved to make Infrastructure as code simple. These tools further abstracted away the code complexity using their domain-specific languages. At the backend, it uses cloud-specific API calls to provision and manage the resources. It helps sysadmins and engineers without programming knowledge to adopt infrastructure as code.

**🚀 Benefits of IaC**

Following are some of the key benefits of IaC

1. With IaC, you can recreate any complex infrastructure with one click.
2. You can version control your infrastructure state in the form of IaC.
3. Developer-centric workflow in infrastructure management. Like developing applications, a standard practice for IaaC code is to follow all standard coding practices like testing, review, etc. Many companies follow test-driven IAC developed to have foolproof infra-change systems.

**🛠️ IaC Tools**

IaaC tools can help you automate and manage all infrastructure components like networks (VPC, Subnets, VPNs, Route tables, etc), servers, cloud-managed services, applications, firewalls, cloud & on-prem managed services, etc.The popular IaC tools are Terraform, Pulumi, Ansible, Chef, and Puppet. Also, there are cloud-specific IaC services like Cloudformation, AWS CDK, etc.

All these IaC tools primarily fall under two categories.

1. Infrastructure Provisioning tools (Terraform, Cloudformation, etc)
2. Configurations management tools (Ansible, Chef, Puppet, etc)

The primary goal of IaC tools is to bring the infrastructure component to the desired state declared by the user. If someone makes a manual change to the resource created by an IaC tool, you can re-run the code and bring it back to the desired state. Once you have the infrastructure code ready, you can use it to create an environment anytime you want without much manual intervention. Just the parameters would change, and the code remains the same.

Most of the IaaC (Open Source) tools can be used on any cloud platform or on-prem environment s without a vendor lock-in unless you use a cloud or vendor-specific tool to manage your infrastructure.

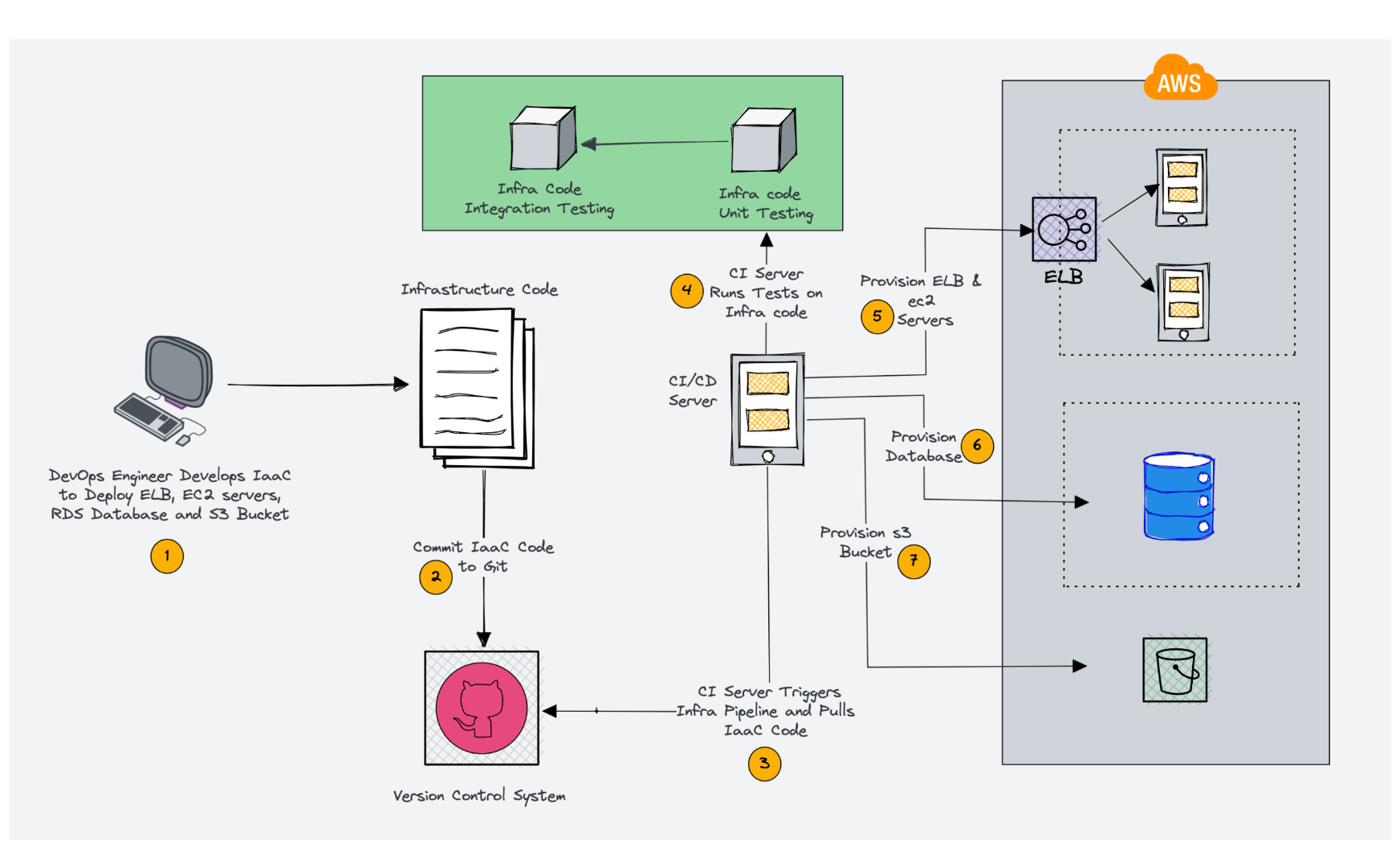
**🧘 Idempotency**

All the **IaC tools** follow the concept of **idempotency**. Meaning, no matter how many times you run the code, if the infrastructure or configuration is already present, it won’t make any change.

For example, you created two servers using Terraform. If you re-run the same terraform code again, it won’t make any changes. However, suppose you manually delete one server and re-run the terraform code. In that case, it will create only one manually deleted server and maintain**the state** of two servers declared by the user in the code.

**🏋️ IaC High-Level Workflow**

The following image shows a high-level overview of **Infrastructure as code development and deployment** workflow. I have used AWS cloud as an example. Here the CI/CD server **orchestrates** the whole provisioning workflow.

[](https://devopscube.com/wp-content/uploads/2022/12/image-18.png)

**What is Infrastructure Provisioning?**

Infrastructure provisioning is the process of **provisioning IT infrastructure resources** like Virtual Servers, Storage, Networking, Cloud managed services, etc.

Terraform, Pulumi and Cloudformation are classic examples of infrastructure provisioning tools. It **creates networks, servers, managed services**, etc. Its primary purpose is to keep the infrastructure in its desired state and reproduce or update it whenever needed. With infrastructure provisioning tools, you can also trigger configuration management tools. So, for example, you can have Terraform code to create Virtual machines and have logic to **run Ansible provisioners** on the created Virtual Machines.

**What is Configuration Management?**

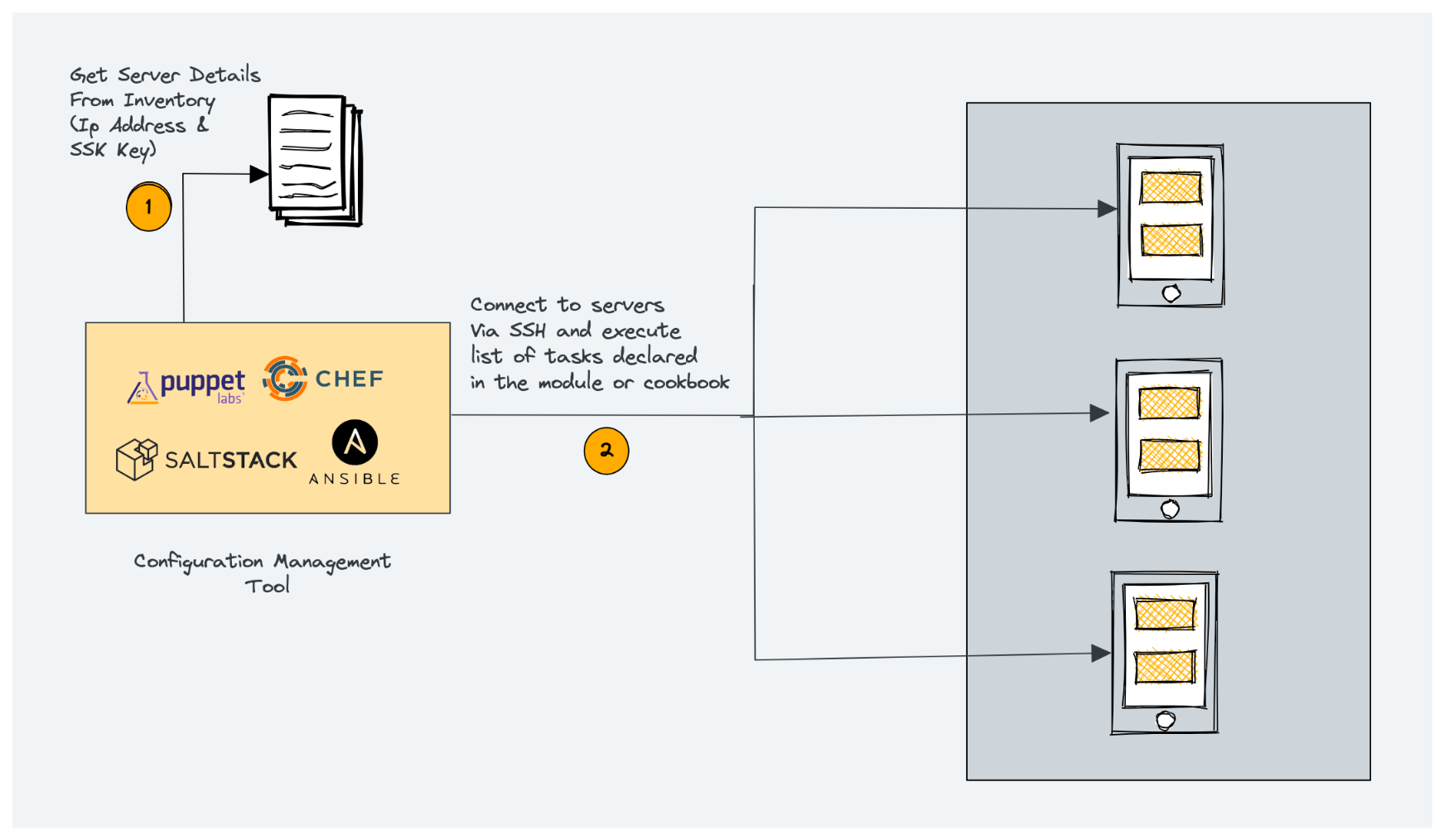
Configuration management is the process of **configuring provisioned infrastructure resources**. For example, configuring a server with required applications or configuring a firewall device.

The primary goal of configuration management tools is to configure the server. Meaning, if you want to automate the installation and configuration of an application (e.g., Nginx) in a server, we use a tool like Ansible and Chef. It does all the configurations in an **idempotent manner**.

Also, these tools help in managing the **configuration drift**. It ensures all the servers are running in the same configuration mentioned in the ansible-playbook or a chef cookbook. In the case of an agent-based chef/puppet, if someone changes the server config manually, the chef agent brings it back to the desired state, as mentioned in the cookbook.

All the configuration management tools keep an inventory of the server’s IP address and SSH credentials to connect to the servers. However, in **cloud environments** where servers are dynamically provisioned, it uses an API-based [dynamic inventory](https://devopscube.com/setup-ansible-aws-dynamic-inventory/) to get the server details.

The following image shows a high-level overview of how a configuration management tool works.

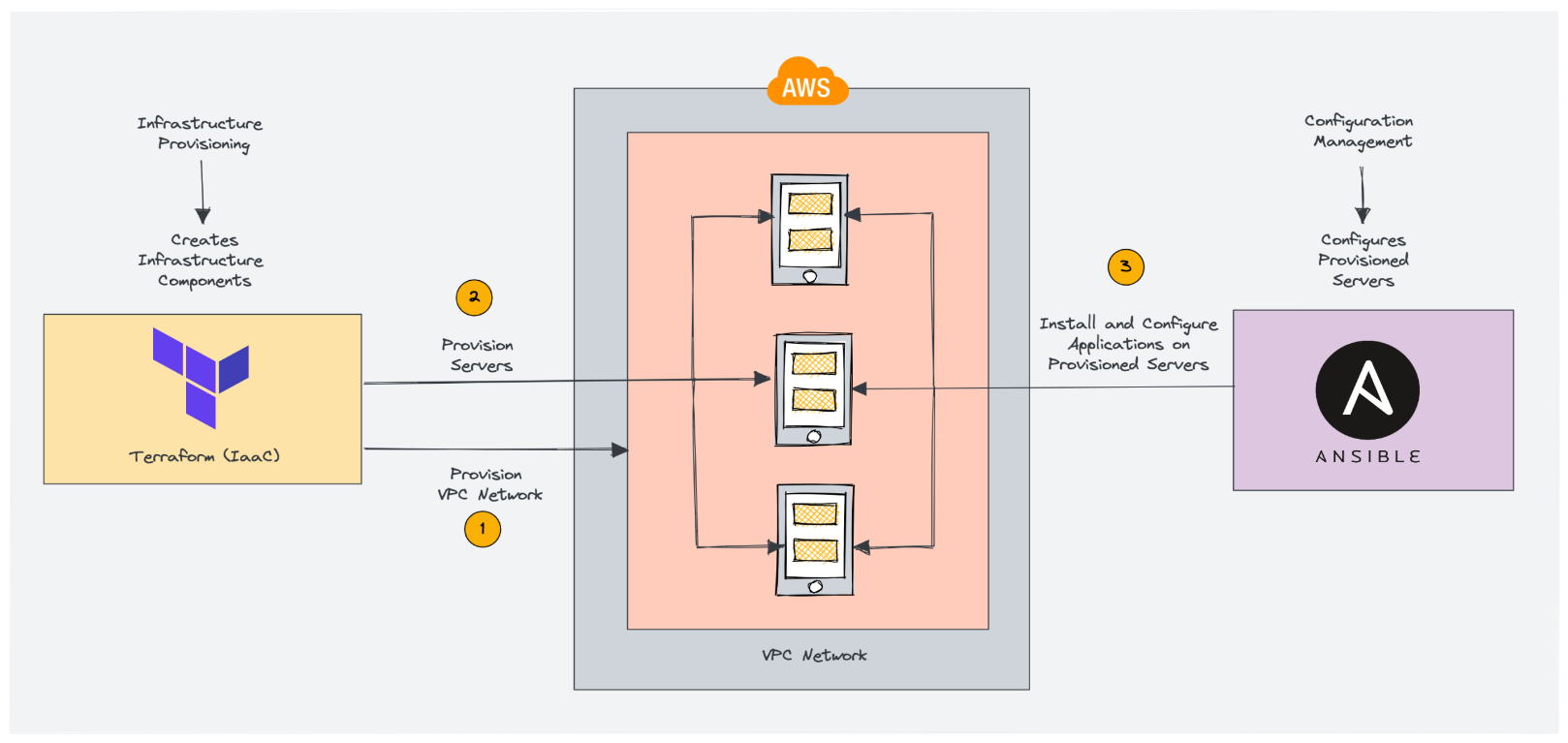
[](https://devopscube.com/wp-content/uploads/2022/12/image-17.png)

Another practical use case of configuration management tools is to create virtual machine images for [immutable infrastructure](https://devopscube.com/immutable-infrastructure/) deployments. For example, tools like [Packer](https://devopscube.com/packer-tutorial-for-beginners/) has provisioner functionality where you can use Ansible, Chef, or Puppet modules to configure the server image with application code.

**Infrastructure Provisioning Vs Configuration Management**

The following image shows the clear difference between Infrastructure Provisioning and configuration management.

1. Infrastructure provisioning tool Terraform is responsible for providing the network and servers.
2. Configuration management tool Ansible configures applications inside servers provisioned by Terraform.

[](https://devopscube.com/wp-content/uploads/2022/12/image-19.png)

**IaC & Configuration Management FAQs**

**Can you use configuration management tools to provision resources?**

Yes. Configuration management tools support provisioning servers using their resources; however, the recommended approach is to use full-fledged infrastructure provisioning tools.

**What is the difference between orchestration and configuration management?**

Orchestration is the process of coordinating multiple automation pipelines and integrating different toolsets. At the same time, configuration management configures a server with the required applications and configurations.

**What is a Practical Example of Orchestration in DevOps?**

One practical example of Orchestration is a Jenkins CI/CD pipeline. When a developer raises a pull request or commits code to git, Jenkins takes care of testing, packaging, creating infrastructure resources, and deploying applications to the server. Here Jenkins orchestrates the whole process integrating multiple DevOps tools and executing the automation in order.