Reference - <https://d1.awsstatic.com/whitepapers/DevOps/infrastructure-as-code.pdf>

Infrastructure as Code

Infrastructure managers have often performed such provisioning manually. The manual processes have certain disadvantages

* Higher cost because they require human capital that could otherwise go toward more important business needs.
* Inconsistency due to human error, leading to deviations from configuration standards.
* Lack of agility by limiting the speed at which your organization can release new versions of services in response to customer needs and market drivers.
* Difficulty in attaining and maintaining compliance to corporate or industry standards due to the absence of repeatable processes.

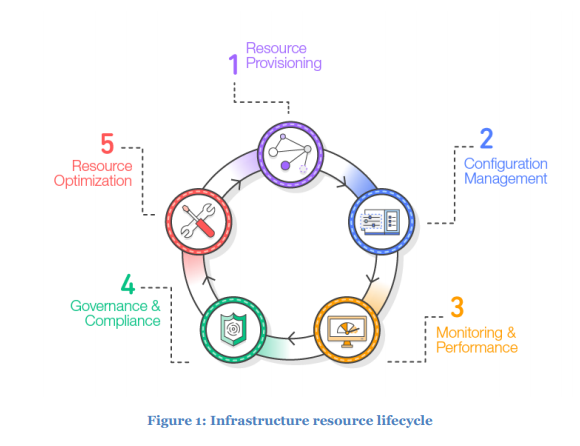
Infrastructure as Code addresses these deficiencies by bringing automation to the provisioning process

infrastructure using configuration files  Infrastructure as Code treats these configuration files as software code.

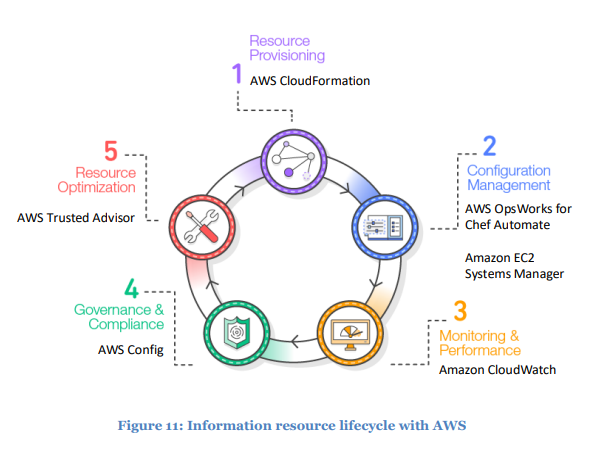
These files can be used to produce a set of artifacts, namely the compute, storage, network, and application services that comprise an operating environment.

Infrastructure as Code eliminates configuration drift through automation, thereby increasing the speed and agility of infrastructure deployments.

**Infrastructure Resource Lifecycle**



1. **Resource provisioning.** Administrators provision the resources according to the specifications they want.
2. **Configuration management.** The resources become components of a configuration management system that supports activities such as tuning and patching.
3. **Monitoring and performance.** Monitoring and performance tools validate the operational status of the resources by examining items such as metrics, synthetic transactions, and log files.
4. **Compliance and governance**. Compliance and governance frameworks drive additional validation to ensure alignment with corporate and industry standards, as well as regulatory requirements.
5. **Resource optimization.** Administrators review performance data and identify changes needed to optimize the environment around criteria such as performance and cost management.



Resource Provisioning

**IAAC address** - need for a repeatable process for instantiating resources consistently

**Usecases :**

* A release manager needs to build a replica of a cloud-based production environment for disaster recovery purposes. The administrator designs a template that models the production environment and provisions identical infrastructure in the disaster recovery location.
* A university professor wants to provision resources for classes each semester. The students in the class need an environment that contains the appropriate tools for their studies. The professor creates a template with the appropriate infrastructure components, and then instantiates the template resources for each student as needed.
* A service that has to meet certain industry protection standards requires infrastructure with a set of security controls each time the service is installed. The security administrator integrates the security controls into the configuration template so that the security controls are instantiated with the infrastructure.
* The manager of a software project team needs to provide development environments for programmers that include the necessary tools and the ability to interface with a continuous integration platform. The manager creates a template of the resources and publishes the template in a resource catalog. This enables the team members to provision their own environments as needed.

AWS Tool – AWS CloudFormation

* create, manage, provision, and update a collection of related AWS resources in an orderly and predictable way.
* AWS CloudFormation uses templates written in JSON or YAML format to describe the collection of AWS resources (known as a stack), their associated dependencies, and any required runtime parameters.
* In effect, you are applying version control to your AWS infrastructure the same way you do with your application code.
* Templates can reference the outputs of other templates, which enables modularization.

Configuration Management

address the ongoing configuration management needs of the environment.

USecases :

* A release manager wants to deploy a version of an application across a group of servers and perform a rollback if there are problems.
* A system administrator receives a request to install a new operating system package in developer environments, but leave the other environments untouched.
* An application administrator needs to periodically update a configuration file across all servers housing an application.

Options :

* Provision new resources and dispose old one (Infrastructure immutability). Eliminates configuration drif
* In environments that have high levels of durability, it might be preferable to have ways to make incremental changes to the current resources instead of reprovisioning them.

EC2 Systems Manager and OpsWorks for chef automate