Infrastructure as Code

* No resources are manually created, which is great for control
* The code can be version controlled for example using git
* Changes to the infrastructure are reviewed through code

Cost

* Each resources within the stack is staged with an identifier so you can easily see how much a stack costs you
* You can estimate the cost of your resources using the CloudFormation template

Productivity

* Ability to destroy and re-create an infrastructure on the cloud on the fly
* Automated generation of Diagram for you templates
* Declarative programming (no need to figure out ordering and orchestration)

Separation of concern: create many stacks for many apps, and many layers

* VPC stacks
* Network Stacks
* App Stacks

How Cloudformation Works

* Templates have to be uploaded in S3 and then referenced in CloudFormation
* To update a template, we can’t edit previous ones. We have to reupload a new version of the template to AWS
* Stacks are identified by a name
* Deleting a stack deletes every single artifact that was created by CloudFormation

Deploying CloudFormation Templates

* Manual way:
  + Editing templates in the CloudFormation Designer
  + Using the console to input parameters, etc
* Automated way:
  + Editing templates in a YAML file
  + Using the AWS CLI to deploy the templates
  + Recommended way when you want to fully automate your flow

**CloudFormation Building Blocks:**

Templates components (one course section for each)

1. Resources: your AWS resources declared in the template (**Mandatory**)
2. Parameters: the dynamic inputs for you template
3. Mappings: the static variables for your template
4. Outputs: References to what has been created
5. Conditionals: List of conditions to perform resource creation
6. Metadata

Template helpers:

1. References
2. Functions