# What is CloudFormation?

* CloudFormation is a declarative way of outlining your AWS Infrastructure, for any resources (most of them are supported)
* For example, within a CloudFormation template, you say:
  + I want a security group
  + I want two EC2 instances using this security group
  + I want an S3 bucket
  + I want a load balancer (ELB) in front of these machines

Then CloudFormation creates those for you, in the right order, with the exact configuration that you specify

# Benefits of Cloud Formation:

**Infrastructure as a Code:**

* No resources are manually created, which is excellent for control
* Changes to the infrastructure are reviewed through code

# Cost:

* Each resources within the stack is tagged with an identifier so you can easily see how much a stack costs you
* You can estimate the costs of your resources using the CloudFormation template
* Savings strategy: In Dev, you could automation deletion of templates at 5 PM and recreated at 8 AM, safely

# Productivity:

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

# Don’t reinvent the wheel

* Leverage existing templates on the web!
* Leverage the documentation

# Supports (almost) all AWS resources:

* Everything we’ll see in this course is supported
* You can use “custom resources” for resources that are not supported

# Problem on AWS for Developers:

* Managing infrastructure
* Deploying Code
* Configuring all the databases, load balancers, etc
* Scaling concerns
* Most web apps have the same architecture (ALB + ASG)
* All the developers want is for their code to run!
* Possibly, consistently across different applications and environments

# Amazon Elastic Beanstalk:

* Elastic Beanstalk is a developer centric view of deploying an application on AWS
* It uses all the component’s we’ve seen before: EC2, ASG, ELB, RDS, etc…
* But it’s all in one view that’s easy to make sense of!
* We still have full control over the configuration
* Beanstalk = Platform as a Service (PaaS)
* Beanstalk is free but you pay for the underlying instances

# Managed Service:

* Instance configuration / OS is handled by Beanstalk
* Deployment strategy is configurable but performed by Elastic Beanstalk
* Capacity provisioning
* Load balancing & auto-scaling
* Application health-monitoring & responsiveness

Just the application code is the responsibility of the developer Three architecture models:

* Single Instance deployment: good for dev
* LB + ASG: great for production or pre-production web applications
* ASG only: great for non-web apps in production (workers, etc..)

Support for many platforms:

* Go
* Java SE
* Java with Tomcat
* .NET on Windows Server with IIS
* Node.js
* PHP
* Python
* Ruby
* Packer Builder
* Single Container Docker
* Multi-Container Docker
* Preconfigured Docker

If not supported, you can write your custom platform (advanced)

# Elastic Beanstalk – Health Monitoring

* Health agent pushes metrics to CloudWatch
* Checks for app health, publishes health events