What is AWS

- AWS is a Cloud Provider
- They provide you with servers and services that you can use on demand and scale easily
- AWS has revolutionized IT over time
- AWS powers some of the biggest websites in the World (for example Netflix)
- Today, AWS has so many services (> 50) that it became hard to manage all the resources manually or within disparate scripts

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 machines using this security group
  - I want two Elastic IPs for these EC2 machines
  - 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 AWS CloudFormation (1/2)

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

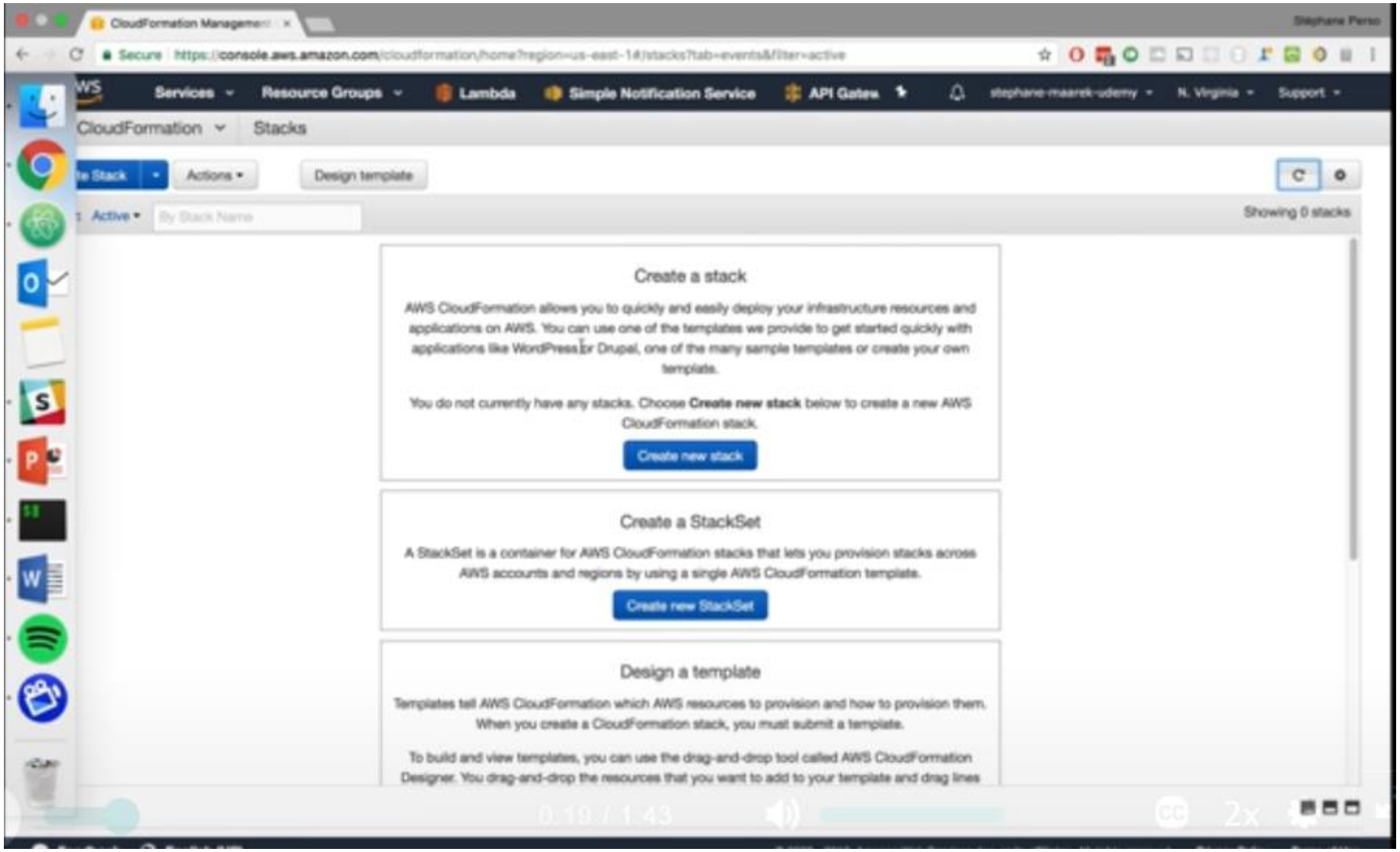
- Cost
  - Each resources within the stack is stagged with an identifier so you can easily see how much a stack costs you
  - You can estimate the costs of your resources using the CloudFormation template
  - Savings strategy: In Dev, you could automation deletion of templates at 5 PM and recreated at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

Benefits of AWS CloudFormation (2/2)

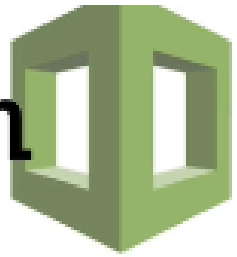
- Productivity
  - Ability to destroy and re-create an infrastructure on the cloud on the fly
  - Automated generation of Diagram for your templates!
  - Declarative programming (no need to figure out ordering and orchestration)

- Separation of concern: create many stacks for many apps, and many layers. Ex:
  - VPC stacks
  - Network stacks
  - App stacks

- Don't re-invent the wheel
  - Leverage existing templates on the web!
  - Leverage the documentation


CloudFormation Management
Secure https://console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks?tab=events&filter=active
Services
Resource Groups
Lambda
Simple Notification Service
API Gatew
stephane-maarek-udemy
N. Virginia
Support
CloudFormation
Stacks
Actions
Design template
Active
By Stack Name
Showing 0 stacks
Create a stack
AWS CloudFormation allows you to quickly and easily deploy your infrastructure resources and applications on AWS. You can use one of the templates we provide to get started quickly with applications like WordPress or Drupal, one of the many sample templates or create your own template.
You do not currently have any stacks. Choose Create new stack below to create a new AWS CloudFormation stack.
Create new stack
Create a StackSet
A StackSet is a container for AWS CloudFormation stacks that lets you provision stacks across AWS accounts and regions by using a single AWS CloudFormation template.
Create new StackSet
Design a template
Templates tell AWS CloudFormation which AWS resources to provision and how to provision them. When you create a CloudFormation stack, you must submit a template.
To build and view templates, you can use the drag-and-drop tool called AWS CloudFormation Designer. You drag-and-drop the resources that you want to add to your template and drag lines
0:19 / 1:43
2x

CloudFormation vs Ansible / Terraform



- CloudFormation is AWS native, and will always contain the latest features and options for AWS Services
- CloudFormation is state based, and AWS figures out how to reach that state
- Ansible and Terraform are instruction based, and it can be difficult to fully orchestrate your stacks
- Ansible and Terraform have to be updated every time a new Services or API option comes from AWS, which can take a long time

- I have used Ansible and CloudFormation, and for AWS related work, I heavily recommend CloudFormation.

Introductory Example

- We're going to create a simple EC2 instance.
- Then we're going to create to add an Elastic IP to it
- And we're going to add two security groups to it
- For now, forget about the code syntax. We'll do a much bigger deep dive later on

Summary of 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 re-upload 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.

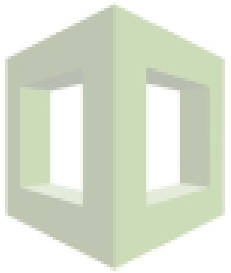
YAML Crash Course

```yaml
invoice:        34843
date   :        2001-01-23
bill-to:
    given  :    Chris
    family :    Dumars
    address:
        lines: |
            458 Walkman Dr.
            Suite #292
        city    : Royal Oak
        state   : MI
        postal  : 48046
product:
    - sku         : BL394D
      quantity    : 4
      description : Basketball
      price       : 450.00
    - sku         : BL4438H
      quantity    : 1
      description : Super Hoop
      price       : 2392.00
```

- YAML and JSON are the languages you can use for CloudFormation.
- **JSON is horrible for CF**
- **YAML is great in so many ways**
- Let's learn a bit about it!

- Key value Pairs
- Nested objects
- Support Arrays
- Multi line strings

Hands-On
Creating a S3 bucket



- Creating a S3 bucket is free
- S3 is the AWS Service for storing static files in a replicated and globally available way
- It powers many websites, Single Page Apps, hosts all the Netflix video content, etc.

- We'll use CloudFormation to provision a S3 bucket!

Understanding the CloudFormation template options

Let's learn about the parameters that are common to any CloudFormation template

1. Tags
2. Permissions
3. Notification Options
4. Timeouts
5. Rollback on Failure
6. Stack Policy

Using the CloudFormation Designer

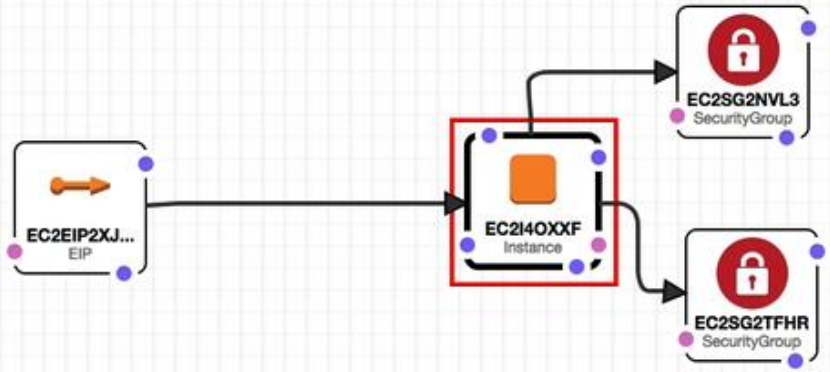
- The CloudFormation Designer can really help visualize a CloudFormation Stack
- It's also very handy to just quickly draft a CloudFormation template
- It has strong capabilities to verify that your template is also well written

Close

Resource types

- NetworkInterface
- PlacementGroup
- Route
- RouteTable
- SecurityGroup
- SpotFleet
- Subnet
- SubnetCidrBlock
- VPC
- VPCCidrBlock
- VPCEndpoint

File: 'new.template'



Properties | Metadata | CreationPolicy | DeletionPolicy | DependsOn | Condition

MyE

Choose

```
1 Resources:
2   EC2I4OXXF:
3     Type: 'AWS::EC2::Instance'
4     Properties:
5       SecurityGroupIds:
6         - !Ref EC2SG2TFHR
7         - !Ref EC2SG2NVL3
8
```

CloudFormation Building Blocks

Templates components (one course section for each):

1. Resources: your AWS resources declared in the template
2. Parameters: the dynamic inputs for your 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

Templates helpers (learning as we encounter them):

1. References
2. Functions

Deploying CloudFormation templates

- Manual way:
  - Editing templates in the CloudFormation Designer
  - Using the console to input parameters, etc
  - We'll mostly do this way in the course for learning purposes

- Automated way:
  - Editing templates in a YAML file
  - Using the AWS CLI (Command Line Interface) to deploy the templates
  - We'll learn about it in the advanced section of the course

What are parameters?

- Parameters are a way to provide inputs to your AWS CloudFormation template
- They're important to know about if:
  - You want to reuse your templates across the company
  - Some inputs can not be determined ahead of time
- Parameters are extremely powerful, controlled, and can prevent errors from happening in your templates thanks to types.

When should you use a parameter?

- Ask yourself this:
  - Is this CloudFormation resource configuration likely to change in the future?
  - If so, make it a parameter.

- You won't have to re-upload a template to change its content ☺

Parameters
Theory & Hands-On

Parameters can be controlled by all these settings:

- Type:
  - String
  - Number
  - CommaDelimitedList
  - List<Type>
  - AWS Parameter (to help catch invalid values – match against existing values in the AWS Account)
- Description
- Constraints

Parameters
Theory & Hands-On

Parameters can be controlled by all these settings:

- Type:
  - String
  - Number
  - CommaDelimitedList
  - List<Type>
  - AWS Parameter (to help catch invalid values – match against existing values in the AWS Account)
- Description
- Constraints
- ConstraintDescription (String)
- Min/MaxLength
- Min/MaxValue
- Defaults
- AllowedValues (array)
- AllowedPattern (regexp)
- NoEcho (Boolean)

What are resources?

- Resources are the core of your CloudFormation template.
- They represent the different AWS Components that will be created and configured
- Resources are declared and can reference each other

- AWS figures out creation, updates and deletes of resources for us
- There are over 224 types of resources (!)
- Resource types identifiers are of the form:

```
AWS::aws-product-name::data-type-name
```

What are resources?

- Resources are the core of your CloudFormation template.
- They represent the different AWS Components that will be created and configured
- Resources are declared and can reference each other

- AWS figures out creation, updates and deletes of resources for us
- There are over 224 types of resources (!)
- Resource types identifiers are of the form:

```
AWS::aws-product-name::data-type-name
```