Main Architecture Overview and CFN Template.

- In this lecture we are going to start building our base architecture, we are gonna be using a nested architecture, or, a template file that reference other template files, we do this to have a modular cloudformation definition.

Lets find the main parts and build our main architecture template:

Going from inside to outside we have:
* Network & Security: where we define security groups, VPC
* Load Balancer: where we define all the load balancer settings
* Compute Area - ECS Cluster: here we will have the ECS cluster configuration
* Service & Task Definition: for the ECS
* Database: we will define the DB cluster here

Knowing this key points, lets start building our yaml file to reflects all. Here comes a tip, everytime I start to build a template file I follow this architecture:

Description: We use the description part to explain about our cloudformation template. Because this is the main template is a nice practice to include all the architecture details here, services, architecture type, etc

Parameters: we use the parameters section to pass values into our template resources. With parameters, we are able to create templates that are customized each time we create a stack. Each parameter must contain a value. And also we can specify a default value to make the parameter optional so that you don't need to pass in a value when creating a stack.

Optional Metadata: we are going to be using the metadata fields to guide the users during the cloudformation creation.

Resources: The Resources section declares the AWS resources that you want to include in the stack, such as an Amazon EC2 instance or an Amazon S3 bucket. We must declare each resource separately; however, if we have multiple resources of the same type, we can declare them together by separating them with commas.

Outputs: declares output values that you can import into other stacks (to create cross-stack references), return in response (to describe stack calls), or view on the AWS CloudFormation console. For example, you can output the S3 bucket name for a stack to make the bucket easier to find.

Following this 4 sections, lets analyzes the parameters needed to start:
First we should need the repository where everything resides (Code, Dockerfiles, images, etc).
Second will be the default repository branch and
Third our Database Password.

The codeCommitRepo variable should be a String and we are going to specify a description for this parameter. This will be our codecommit repository name, so we type: CodeCommit Repository Name.

The next parameter is the RespositoryBranch, this one is also a String and the description will be: CodeCommit Reporsitory Branch.

The last parameter is DbPassword, this one is also a String and lets use the description field: Backend DB Password.

Lets continue with our template:
- metadata: the metadata resource type is AWS::Cloudformation::Interface this is a metadata key that defines how parameters are grouped and sorted in the AWS CloudFormation console. Normally, when you create or update stacks in the console, the console lists input parameters in alphabetical order by their logical IDs. By using this key, you can define your own parameter grouping and ordering so that users can efficiently specify parameter values.

In addition to grouping and ordering parameters, you can define labels for parameters. A label is a friendly name or description that the console displays instead of a parameter's logical ID. Labels are useful for helping users understand the values to specify for each parameter.

We define the label with the key ParameterLabels and we are going to use the first 2 parameters we define before:

First CodeCommitRepo, we use the tag “default” to specify the message. As we add in the previews description we will type CodeCommit Repository Name.

Next RepositoryBranch, using the default tag we specify: CodeCommit Repository Branch Name (master).

Remember we are going to be launching this template to build a stack through the aws console so its nice for us to group the set up parameters during the beginning to help even more the people, and for that we use the tag “ParameterGroups”. The ParameterGroups tags give us the option to label the group, we use the same default for that and specify: CodeCommit Repository Configuration, lastly we specify which parameters are we going to have in the group, because we are going to group everything related with the CodeCommit, we will have the CodeCommitRepo and the RepositoryBranch and with this we finnish our parameters in this template.

Now we are going to define our resources, as we talked before, we will have 5 main resources: Cluster, LoadBalancer, Service, VPC, Database.

Lets start with the VPC, because is the one without dependencies from other resources. First in VPC we are going to define the type, we use the AWS:Cloudformation::Stack resource type that allow us to nest a stack as a resource in this top-level template.

Next we define the properties needed for this resource, first, the templateURL, this one specify The URL of a template that you want to create as a resource. The template must be stored on an Amazon S3 bucket, so the URL must have the form: https://s3.amazonaws.com/

Lastly we will need the parameters needed for this resource to work, because we are building a VPC here we will need the IPv4 CIDR block for the whole network, a couple of subnets with an IPv4 CIDR block and of course we will need a name for this VPC resource.

To do this we declare a Name for this VPC, we will use a Pseudo Parameters for this purpose, The Pseudo parameters are parameters that are predefined by AWS CloudFormation. You do not declare them in your template. Use them as the argument for the Ref function.

The pseudo parameter we are going to use here is AWS::StackName, and as I said we need the Ref function to use it, so we type here: !Ref AWS::StackName. With this we get the name of the stack and we are going to use it in the VPC.

Lets now define our LoadBalancer resource, this one will be also a nested stack, so we will have the same structure, but slightly different, because the load balancer will need parameters from the VPC resource once this one is created. So the first part is the same, this will be a AWS:Cloudformation::Stack type, in properties, we will have the TemplateURL and in parameters starts the interesting part, in our loadbalancer we will need the VpcId or VPC logical id and the Subnets list.

We will do this getting the results from the VPC resource, We will need to get the specific attribute from the VPC output or result, we use for that the !GetAll VPC.Outputs.Subnets and will do the same for the VPCId !GetAll VPC.Outputs.VpcId

We will follow the same process for the Service & the cluster, Service will need the cluster name parameter from the cluster resource and the targetgroup parameter from the loadbalancer. Cluster will need SecurityGroup from the loadbalancer resource and Subnet list, VpcID and subnet 1 & 2 and the VPC security group from the VPC resource output.

Finally we have the Database this one will need the TargetGroup from the LoadBalancer, the dbpassword from the setup parameters, the Vpc Logical Id, Subnet 1 & 2, and Availability Zone 1 & 2.

Our last key point is the output, here we specify the general stack output, here we will need our LoadBalancer URL with this we can take Route53 and make a hosted zone pointing to this url, and we are ready to serve from a domain or subdomain.

In the next videos we are going to explore in deep each of this nested templates.