**7. Understanding Terragrunt core features**

--- **We can keep our terraform code dry**

* What is the meaning of that? We will use terragrunt to manage multiple environments. if I want to use same Terraform configuration file, the right solution is terragrunt. the grant go dry in the sense I should not touch my Terraform configuration files for each and every environment which I want to apply so that I can pass the values from terragrunt according to my environment.
* Let us say it is a qa environment, it is a production environment, whether it is staging environment. What are the environments you are using in your organization? So based on that, I must pass the values from terragrunt, but I should not modify any code or any configuration file in my TerraForm. that is a major point. We must keep our TerraForm code dry.

--- **we can keep our backend configuration dry**

* For example, let us say back-end configuration, back end in the sense instead of storing your Terraform State file in the local system, we use some concept called back-end configuration, back in configuration in the sense I'm using TerraForm State file to be stored in a remote location like S3 Bucket for AWS or for any other target provider.
* Instead of storing our TerraForm State files in our local system, we are storing our TerraForm State files in a remote location.
* Now the problem is, if we are using multiple environments, I must write multiple TerraForm State files to be stored in my S3 bucket. I must store my TerraForm States file individually for every environment. With the help of terragrunt, I can inject some particular back end configuration into my Terraform configuration file before applying the changes on the target provider.

--- **we can add hooks (pre-hooks, post hooks and error hooks)**

* hooks are nothing but I want to trigger something before doing some actions, before creating something, I want to do something, before destroying something. After creating something. I want to trigger some tasks or if I have some error, I want to spin up something or I want to do some clean-ups. Those are called hooks.
* We have something called pre hooks, post hooks and error hooks.

--- **we can work with multiple AWS accounts**

* In that scenario, we can use multiple accounts at a time to manage with the help of roles. We have a concept called cross account access in a IAM service.
* We have something called cross account access, terragrunt will use that feature in order to use the cross-account access and manage multiple accounts for resource creation.
* Let us say for example, I am using one AWS account for production environment, one AWS account for dev environment, when I want my direct grant to communicate to these many accounts, I can use a simple concept called roles. That is what they are talking about in this point.

--- **Inputs and locals**

* terragrunt must pass some values to your variables in your Terraform. Then what it will use? It will use a concept called inputs.
* Locals are something where, in a terragrunt configuration file, I want to define some variable and that must be used. I will define some values in my variables that you can define under local block.

--- **caching**

* So, terragrunt by default will store your terraform configuration files under this cache folder. If you want to do some CleanUp, we can clean up that folder

--- **AWS auth**

* The purpose of this is authentication, If I want to store my credentials or IAM policies and all, then we can use something called AWS auth. Terragrunt is providing this feature for AWS.

--- **Debugging**

* Terraform and terragrunt usually show you the information what is happening. Sometimes you want to find out what is happening behind the scenes. For this we must use something called debug mode of terragrunt.

--- **lock file handling**

* Because when you are applying some terraform, I am talking about Terraform right now. when you run some Terraform plan, apply destroy. When it is running it, it will read your Terraform State file. If the resource has been already created.
* Now when someone is reading that Terraform State file, I do not want other process to open that file. I do not want a parallel job to open that file and read the information.