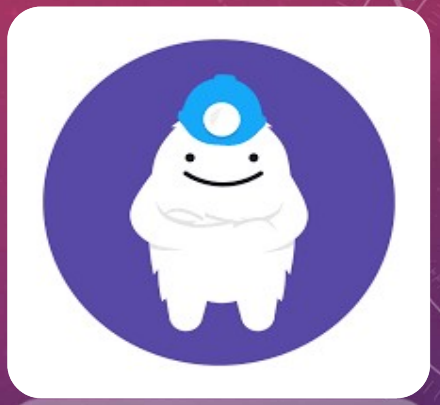
Why we need Terragrunt for Terraform?



- Terragrunt is a thin wrapper for Terraform to keep your configurations DRY
- Managing the Infrastructure for multiple environments is complex, even with Terraform workspace and Modules
- It helps to create Infrastructure for multiple environments from single source, instead of multiple copy of configuration files for each environment
- We can manage remote state for each environment
- Terragrunt inherits all the built-in commands of Terraform and some more options as well
- Terragrunt can pull Terraform configuration code stored in local, remote or any source repository (git)
- It is a opensource tool, which helps to contribute or make development changes as per client requirement

Terraform workspace Vs Terragrunt

```
resource "aws_instance" "web" {
  ami                         = data.aws_ami.ubuntu_ami.id //ami-0c2b8ca1dad447f8a
  instance_type               = var.inst_type
  availability_zone           = var.az_name
  count                       = terraform.workspace == "default" ? 3 : 1
  user_data                   = file("nginx-install.sh")
  key_name                    = "sudhams_virginia_demo"
  associate_public_ip_address = true
  vpc_security_group_ids      = [aws_security_group.allow_http_ssh.id]

  tags = {
    Name = "Nginx-application-${terraform.workspace}"
  }
}
```

```
#Show current workspace
terraform workspace show

#List workspaces
terraform workspace list

#select workspace to switch to dev from default
terraform workspace select dev

#Create Resource for specified environment workspace
terraform apply [-auto-approve]

#Delete resources from default workspace
terraform destroy [-auto-approve]

#verify if all resources related to default workspace are cleaned up
```

```
resource "aws_instance" "web" {
  ami                         = data.aws_ami.ubuntu_ami.id //ami-0c2b8ca1dad447f8a
  instance_type               = var.inst_type
  count                       = var.inst_count
  availability_zone           = var.az_name
  user_data                   = file("nginx-install.sh")
  associate_public_ip_address = true
  vpc_security_group_ids = [aws_security_group.allow_http_ssh.id]

  tags = {
    Name = local.environment-name
  }
}
```

```
terraform {
    # Deply version @v0.1.0 on target provider
    source = "git::git@github.com:sudheerdemo/modules.git//app?ref=v0.1.0"
}

inputs = {
    inst_count = 3
    inst_type  = "t2.large"
    az_name    = "us-east-1a"
    vpc_id     = dependency.vpc.outputs.vpc_id
}
```

Terragrunt Workflow (apply)