Module Checklist

Infrastructure as Code with Terraform

By Techworld with Nana

Video Overview

- ★ Introduction to Terraform
- ★ Install Terraform & Local Setup
- ★ Providers
- ★ Resources & Data Sources
- ★ Change/Destroy Resources
- ★ More Terraform commands
- ★ Terraform State
- ★ Terraform Output
- ★ Variables
- ★ Environment Variables
- ★ Initialize Git Repository
- ★ Demo Project 1: Automate your AWS Infrastructure - Part 1
- ★ Demo Project 1: Automate your AWS Infrastructure - Part 2
- ★ Demo Project 1: Automate your AWS Infrastructure - Part 3
- ★ Provisioners
- ★ Modules - Part 1
- ★ Modules - Part 2
- ★ Modules - Part 3
- ★ Demo Project 2: Terraform & AWS EKS - Part 1
- ★ Demo Project 2: Terraform & AWS EKS - Part 2
- ★ Demo Project 2: Terraform & AWS EKS - Part 3
- ★ Demo Project 3: Complete CI/CD with Terraform - Part 1
- ★ Demo Project 3: Complete CI/CD with Terraform - Part 2
- ★ Demo Project 3: Complete CI/CD with Terraform - Part 3
- ★ Terraform Remote State

Video Overview

Demo Projects	
Git Project	https://gitlab.com/nanuchi/terraform-learn

Check your progress... 1/11

Introduction to Terraform

- ❏ Watched video

Install Terraform & Local Setup

- ❏ Watched video
- ❏ **Demo executed - Install Terraform:**
  - ❏ Terraform installed
  - ❏ “terraform” project created

Useful Links:

- Guide to install Terraform for different OS:
  https://learn.hashicorp.com/tutorials/terraform/install-cli
  https://www.terraform.io/downloads.html
- Visual Studio Code Installation: https://code.visualstudio.com/download

Providers

- ❏ Watched video
- ❏ **Demo executed:**
  - ❏ Use AWS Provider

Useful Links:

- Browse Terraform Providers: https://registry.terraform.io/browse/providers
- Project: https://gitlab.com/nanuchi/terraform-learn/-/tree/master

Check your progress... 2/11

Resources and Data Sources

- ❑ Watched video
- ❑ **Demo executed**
  - ❑ Created new VPC
  - ❑ Created Subnet in that new VPC
  - ❑ Created new Subnet in existing default VPC (with data)

Useful Links:

- Project: https://gitlab.com/nanuchi/terraform-learn/-/tree/master

Change and destroy resources

- ❑ **Watched video**
- ❑ **Demo executed :**
  - ❑ added tags to existing resources
  - ❑ removed tag
  - ❑ destroyed a resource

Useful Links:

- Project: https://gitlab.com/nanuchi/terraform-learn/-/tree/master

More terraform commands

- ❑ **Watched video**
- ❑ **Demo executed :**
  - ❑ Executed preview command
  - ❑ Applied config file without preview
  - ❑ Destroyed complete infrastructure

Useful Links:

- Project: https://gitlab.com/nanuchi/terraform-learn/-/tree/master

Check your progress... 3/11

Terraform State

- ❑ Watched videos
- ❑ **Demo executed**

Useful Links:

- Project: https://gitlab.com/nanuchi/terraform-learn/-/tree/master

Terraform Output

- ❑ Watched video
- ❑ **Demo executed - define output values**

Useful Links:

- Project: https://gitlab.com/nanuchi/terraform-learn/-/tree/master

Variables

- ❑ Watched video
- ❑ **Demo executed:**
  - ❑ Passed variables in 3 different ways
  - ❑ Restricted value of variable by defining a type

Useful Links:

- Everything about Input Variables: https://www.terraform.io/docs/configuration/variables.html
- Project: https://gitlab.com/nanuchi/terraform-learn/-/tree/master

Check your progress... 4/11

Environment variables

- ❑ Watched video
- ❑ **Demo executed:**
  - ❑ Used environment variables to extract AWS credentials
  - ❑ Set variable using *TF_VAR_name* environment variable

Useful Links:

- Custom Environment variables: https://www.terraform.io/docs/commands/environment-variables.html
- Project: https://gitlab.com/nanuchi/terraform-learn/-/tree/master

Initialize Git Repository

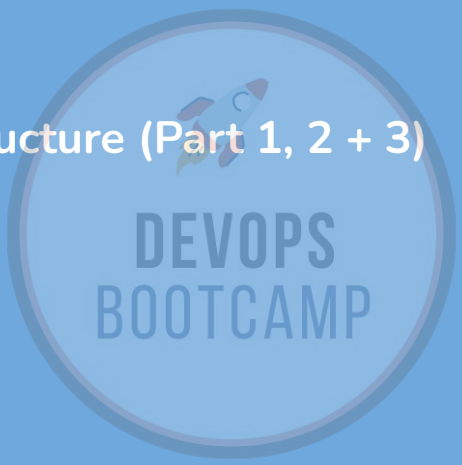
- ❑ Watched video
- ❑ **Demo executed:**
  - ❑ Created Remote Git Repository for Terraform Configuration Files
  - ❑ Connected remote Git Repository with local project
  - ❑ Added .gitignore files

Best Practices so far:

- **Security:** Don't include sensitive data in the Terraform configuration file! Because it will be checked in in your git repository.
- Use terraform apply with the configuration file to make infrastructure changes, instead of executing commands directly. Especially when you work in a team. Because otherwise, infrastructure's current state and the desired state represented in the configuration file do not correspond anymore!

Check your progress... 5/11

Terraform & AWS



Demo Project 1: Automate AWS Infrastructure (Part 1, 2 + 3)

- ❏ Watched video
- ❏ **Demo executed:**
  - ❏ Created VPC & Subnet
  - ❏ Created custom Route Table
  - ❏ Added Subnet Association with Route Table
  - ❏ Configured Default/Main Route Table
  - ❏ Created Security Group
  - ❏ Configured Default Security Group
  - ❏ Created EC2 Instance (Fetch AMI, Create ssh key-pair and download .pem file and restrict permission)
  - ❏ SSH into EC2 instance
  - ❏ Configured ssh key pair in Terraform config file
  - ❏ Created EC2 Instance
    - Fetch AMI
    - Create ssh key-pair and download .pem file
    - restrict permission
  - ❏ SSH into EC2 instance
  - ❏ Automated ssh key-pair - configured ssh key pair in Terraform config file
  - ❏ Configured Terraform to install Docker and run nginx image
  - ❏ Extract shell commands to own shell script
  - ❏ Accessed nginx through Browser

Check your progress... 6/11

Demo Project 1: Automate AWS Infrastructure (Part 1, 2 + 3)

Useful Links:

- Project Repo - Provision EC2 with new components: https://gitlab.com/nanuchi/terraform-learn/-/tree/feature/deploy-to-ec2
- Project Repo - Provision EC2 with default components: https://gitlab.com/nanuchi/terraform-learn/-/tree/feature/deploy-to-ec2-default-components
- EC2 Instance Resource: https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance
- Data Sources Filtering: https://registry.terraform.io/providers/hashicorp/oci/latest/docs/guides/filters
- Generate a new ssh key: https://www.ssh.com/ssh/keygen/

Best Practices:

- With Terraform: Create own VPC and leave the defaults created by AWS as is
- **Security**: Store your .pem file ssh private key in .ssh folder. Restrict permission (only read for our User) on .pem file
- **Security**: Don't hardcode public_key in Terraform config file!

Check your progress... 7/11

Provisioners

- ❏ Watched video
- ❏ **Demo executed:**
  - ❏ Used “remote-exec” provisioner
  - ❏ Used “file” provisioner
  - ❏ Used “local-exec” provisioner

Useful Links:

- Project Repo: https://gitlab.com/nanuchi/terraform-learn/-/tree/feature/provisioners

Best Practices:

- Use configuration management tools instead of Terraform provisioners

Modules (Part 1, 2, 3)

- ❏ Watched videos
- ❏ **Demo executed:**
  - ❏ Extracted output values, variables and providers into its own file
  - ❏ Created subnet module and used it in root config file
  - ❏ Created webserver module and used it in root config file
  - ❏ Executed terraform apply successfully

Useful Links:

- Module Creation - Recommended Pattern: https://learn.hashicorp.com/tutorials/terraform/pattern-module-creation?in=terraform/modules
- Project Repo: https://gitlab.com/nanuchi/terraform-learn/-/tree/feature/modules

Best Practices:

- Terraform Project Structure: Own .tf file for providers, variables, data sources and output values
- Modules: encapsulate configuration into distinct logical components

Check your progress... 8/11

Terraform & AWS EKS

Demo Project 2: Terraform & AWS EKS (Part 1, 2 & 3)

- ❏ Watched videos
- ❏ **Demo executed:**
  - ❏ Created the VPC by using the VPC module
  - ❏ Created the EKS cluster and worker nodes by using the EKS module
  - ❏ Configured Kubernetes provider to authenticate with K8s cluster
  - ❏ Applied configurations
  - ❏ Deployed nginx Application/Pod
  - ❏ Terraform destroy (IMPORTANT: delete all your components, if you don't want to get charged for a running cluster!)

Useful Links:

- Project Repo: https://gitlab.com/nanuchi/terraform-learn/-/tree/feature/eks
- VPC Module: https://registry.terraform.io/modules/terraform-aws-modules/vpc/aws/latest
- EKS Cluster Module: https://registry.terraform.io/modules/terraform-aws-modules/eks/aws/latest
- Kubernetes Provider: https://registry.terraform.io/providers/hashicorp/kubernetes/latest/docs

Check your progress... 9/11

Terraform & Jenkins

Demo Project 3: CI/CD with Terraform (Part 1, 2 & 3)

- ❏ Watched videos
- ❏ **Demo executed:**
  - ❏ Created SSH key pair for EC2 Instance
  - ❏ Created Credential in Jenkins
  - ❏ Installed Terraform inside Jenkins Container
  - ❏ Created Terraform configuration files to provision an ec2 server
  - ❏ Created entry-script.sh file to install docker, docker-compose and start containers through docker-compose command
  - ❏ Adjusted Jenkinsfile to include provision and deployment stage
  - ❏ Included docker login to be able to pull Docker Images from private Docker repository
  - ❏ Executed CI/CD pipeline successfully

Useful Links:

- Project Repo: https://gitlab.com/nanuchi/java-maven-app/-/tree/feature/sshagent-terraform
- Install Terraform: https://learn.hashicorp.com/tutorials/terraform/install-cli
- Install docker-compose: https://docs.docker.com/compose/install/
- Terraform environment variables: https://www.terraform.io/docs/commands/environment-variables.html

Best Practice:

- Include TF configuration files in your project folder

Check your progress... 10/11

Demo Project 3: CI/CD with Terraform (Part 1, 2 & 3)

Useful Commands

- Install Terraform in Jenkins:

```
# add HashiCorp key
curl -fsSL https://apt.releases.hashicorp.com/gpg | apt-key add -

# install apt-add-repo command
apt-get install software-properties-common

# add the official HashiCorp Linux repository
apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com
$(lsb_release -cs) main"

# update and install
apt-get update && apt-get install terraform

# verify
terraform -v
```

Check your progress... 11/11

Terraform Remote State

- ❑ Watched video
- ❑ **Demo executed:**
  - ❑ Configured Remote Storage

Useful Links:

- Project Repo: https://gitlab.com/nanuchi/java-maven-app/-/tree/feature/sshagent-terraform
- Backends: https://www.terraform.io/docs/backends/
- Remote State: https://www.terraform.io/docs/state/remote.html
- AWS S3: https://aws.amazon.com/s3/

Best Practice:

- Use Remote Terraform State when working in a team
- Use S3 Bucket Versioning
- **Security**: Enable encryption for the S3 Bucket

More Resources...

More Best Practices

- Use _ (underscore) instead of - (dash) in all resource names, data source names, variable names, outputs etc.
- Only use lowercase letters and numbers
- Use remote state, instead of on your laptop or in Git
- Use a consistent structure and naming convention
- Don’t hardcode values as much as possible - pass as variables or use data sources to get a value