**Day 1**

**Introduction to DevOps**

* Why of DevOps?
* DevOps Tools – Overview and Use case
* Source Control Management (SCM Tools)
* Continuous Integration, Continuous Delivery and Continuous Deployment
* Static Code Analysis Tool
* What are Storage Artifacts?
* What is Configuration Management?

**Introduction to Puppet**

* Overview of Puppet
* Puppet Pre-install tasks
* Hands-on: Puppet Installation and Configuration on Linux
* Hands-on: Puppet Server post-installation configuration
* Overview of Puppet Console
* Puppet Enterprise vs Opensource
* Puppet vs Chef vs Ansible vs Terraform

Lab: Puppet Installation and Configuration. Lab: Overview of Puppet Console

Lab: Puppet Ad-hoc Commands

Lab: Puppet Facts

Lab: Puppet Server and Agent configuration

**Puppet Architecture**

* Puppet Server and Puppet Agent
* Understand how Puppet reports Facts and the Catalog
* Sizing Puppet Master
* Differences between Monolithic and Split Puppet architectures

**Day 2**

**Understanding version control (Git)**

* Git Overview
* Hands-on: Git in Practice
* Hands-on: Setting Up Your Profile
* Hands-on: Creating a Git Repository
* Hands-on: Creating GitHub account
* Hands-on: Cloning Git Repo
* Hands-on: Changes in the Git Repo
* Hands-on: Commit Git Repo
* Hands-on: Push/Pull Git Repo

**Puppet Modules**

* Modules overview
  + Module structure
  + Module names
  + Files in modules
  + Templates in modules
  + Hands-on: Writing modules
* Common Built in Modules
* Hands-on: Writing Manifest files
* Hands-on: Installing modules from the Forge
* Hands-on: Searching the Forge from CLI and web
* Hands-on: Using the Puppet Module command
* Types and Providers
* Lab: Converting nginx.conf into a template
* Lab: Creating and applying a Puppet Manifest
* Lab: Modules, module path structure, and testing

**Puppet Language**

* How Puppet uses resources for configuration management?
* Developing Puppet resources
* File Serving
* Relationships
* Package / File / Service
* Variables
* Conditional statements
* Built-in resource types
* Description of resources–resource types, titles, and body
* Core resource types
* Inspecting resources
* Hands-on: Basic Linux administrator tasks
  + Managing Packages
  + Managing Services
  + Managing Files & Folders
  + Managing Users

Lab: Developing Puppet resources, applying resources to nodes

**Day 3**

**Learning Classes**

* Understanding Puppet classes
* Hands-on: Manifests with Classes
* Hands-on: Class Inheritance

Lab: Manifests and Classes: Create class definitions, validate class syntax and apply to Puppet nodes

Lab: Converting nginx class into a parameterized class

**Values and data types**

* Strings
* Numbers
* Binary
* Booleans
* Arrays
* Hashes
* Sensitive
* Resource and class references
* Lab: Create manifest using various data types

**Learning Variables**

* Using variables in Puppet
* Writing Puppet code without repeating
* Writing Classes with parameters
* Using the facter tool with modules and classes

Lab: Data Driven Modules - Params and Facts

Lab: Variables and Parameters: learn how to assign variables in a manifest, create classes with parameter

**Hiera**

* Introduction to Hiera
* Use cases of Hiera
* Configuring Hiera
* Hiera best practices

Lab: Installing and Using Hiera

**Day 4**

**Conditional Statements**

* Understand and use these conditional statements:
  + If
  + Unless
  + Case
  + Selector

Lab: Conditional Statements: Using conditional statements in resources and module

**Iteration and Loops**

* Iteration functions
* Declaring resources
* Iteration with defined resource types
* Using iteration to transform data
* Breaking out of the loop
* Lab: Refactor the manifest file to use iteration concepts

**Managing nodes in Puppet Enterprise**

* Hands-on: Adding and removing agent nodes
* Hands-on: Adding and removing agentless nodes
* How nodes are counted
* Hands-on: Running Puppet on nodes
* Hands-on: Grouping and classifying nodes
* Hands-on: Making changes to node groups
* Preconfigured node groups

**Orchestrating Puppet runs, tasks, and plans**

* How Puppet orchestrator works?
* Hands-on: Setting up the orchestrator workflow
* Hands-on: Configuring Puppet orchestrator
* Hands-on: Run Puppet on demand
* Tasks in PE
* Plans in PE

**Day 5**

**Learning Advanced Topics**

* Environments
* Hands-on: Creating a dev environment and adding nodes to it
* Hands-on: Creating custom modules
* Automatic data binding

Lab: Resource ordering: Ensure the correct order of modules and classes

Lab: Defined resource types: Create new resource types

Node Excluson and inclusion based on multiple conditions in .pp file entry point

**Hands-on: Misc Use Case Labs**

* Linux Patching
* Windows Patching
* Set Windows env
* Install and remove Windows features
* Harden windows server
* Scheduled task on Windows
* Create user in Windows
* Create group in Windows

**Testing and Troubleshooting**

* Hands-on: Create Tests for Our Code
* Hands-on: Log locations
* Troubleshooting puppet infrastructure run commands
* Troubleshooting connections between components
* Troubleshooting the databases
* Troubleshooting backup and restore
* Using PRY to Inspect the Puppet Server

**Performance tuning in Puppet Enterprise**

* Number of JRubies
* JVM Heap Size
* Tying Together max-active-instances and Heap Size
* Potential JAVA ARGS settings
* Using the puppet infrastructure tune command

**Custom Application orchestration using Puppet**

* Overview
* Install and configure tomcat
* Deploy WAR files to tomcat server
* Configure deployed application

**Extras:**

* Create custom facts
* Node Excluson and inclusion based on multiple conditions in .pp file entry point
* Puppet graph
* puppet apply --graph --noop manifest.pp
* dot -Tpng /var/lib/puppet/state/graphs/relationships.dot -o relationships.png
* Create Custom Task using Ruby