**DevOps**

|  |
| --- |
| **TECHVISION**  **IT TRAINING AND PLACEMENT SERVICES**  **Website:Techvisionit.com** |

**Course Duration:** 60 hours

**Sessions:** Weekdays; Weekends; Crash Course

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **DevOps Course Outline**  **Course Overview**  This course provides hands on to DevOps – the cultural & professional movement that stresses communication, collaboration, integration & automation in order to improve the flow of work between software developers & IT operations professionals.  **Tools Covered (one tool from every stage of life cycle)**   * + Version Control Tool: Git and GitHub   + Build automation Tool: Maven   + Testing Tool: Junit   + Continuous Integration Tool: Jenkins   + Container movement and Container platform (Image based deployment): Docker   + Configuration Management Tools: Ansible, Puppet, Chef, SaltStack   + Continuous Monitoring Tools: Sensu, Nagios   **Topics**  **DevOps Fundamentals**   * System Development Life Cycle(SDLC) * SDLC Models * Agile Methodology (Backlog, Sprint, Scrum Master )   **Version Control Tool – GIT**  **Git Repository**   * Creating a Git Repository * Git Workflow * Tracking File Changes * Files or directory add to stage * Reset from stage * Ignoring Files in Git * Commit to Repository * Reverting to Earlier Commits * Deleting Files in Git   **GitHub – Cloud Repository**   * Creating a Repository in GitHub * Creating a Repository in GitHub Using SSH * Pulling Commits from GitHub * Collaborating between Local and Remote Repository * Push local Repository to GitHub or remote Repository * Merging File Changes in Git * Issue Tracking in GitHub   **Branching Merging And Rebasing in Git**   * Branching in Git * Merging Branches in Git * Fast Forward and Recursive Merge * Recursive MergePreview * Resolving Merge Conflicts in Git * Stashing in Git * Rebasing in Git * Cloning in Git   **Build Automation with Maven**   * Installing Maven * Understanding the lifecycle and dependencies of Maven * Working with the Project Object Model (POM) * Defining project relationships * Using Maven plugins * Creating a sample project with Maven * Writing tests in Maven * Packaging your app   **Unit testing with Junit**  **Prerequisites**  A basic understand of Java is required. JUnit makes extensive use of annotations. Therefore, Java developers should be comfortable with Java annotations.  **JUnit Intro**   * Rules of unit testing * Obtaining, setup, configure JUnit * Unit test execution   **JUnit API**   * Test case/class * Test methods * Assert methods * Test suite * Test runner   **Software Testing**   * Why unit testing * Types of tests * Types of unit tests * Black vs. white box testing * Code coverage * Testable code * Test-driven development   **Installing and Running Jenkins**   * Downloading and Installing Jenkins * Running Jenkins as a Stand-Alone Application * Initial Configuration   **Job Types in Jenkins**   * Different types of Jenkins Items * Configuring Source Code Management(SCM) * Working with Subversion * Working with Git * Storing Credentials * Service Accounts * Schedule Build Jobs * Polling the SCM * Polling vs Triggers * Maven Build Steps   **Jenkins Plugins**   * Jenkins Plugins - SCM * Jenkins Plugins – Build and Test * Jenkins Plugins – Analyzers * Jenkins for Teams * Installing Jenkins Plugins   **Distributed Builds with Jenkins**   * Agent Machines * Configure Jenkins Master * Configure Projects * Conclusion   **Continuous Delivery and the Jenkins Pipeline**   * Continuous Delivery * Continuous Delivery (cont'd) * DevOps and Continuous Delivery * Continuous Delivery Challenges * Continuous Delivery with Jenkins * The Pipeline Plugin * The Pipeline Plugin (cont'd) * Defining a Pipeline * A Pipeline Example * Pipeline Example (cont'd) * Parallel Execution * Creating a Pipeline * Invoking the Pipeline * Conclusion   **Docker Containerization Boot Camp**  **Introduction**   * What can you use Docker for? * How Docker fits into the development lifecycle * How Docker ensures consistency from development through UAT and staging, and on to production * Example use cases of Docker in the real world   **The components of Docker**   * Underlying technology * Docker client and server * Filesystem images * Registries * Containers * Networking   **Getting set up to start using Docker**   * Getting set up on Windows * Trying out our first container * Getting set up for production on Linux * Tweaking your production environment for best performance   **Container management**   * Container naming * Starting and stopping containers * Attaching to a container * Seeing what is happening in a container * Running a process inside a container * Daemonizing a container * Automatic container restarts * Deleting containers when we are finished with them   **Docker images and repositories**   * Docker images explained * How Docker images work * Getting a list of images * Searching for images on a repository * Pulling an image * Creating our own image * Specify an image in a Dockerfile * Building Dockerfile images * Using the build cache for templating * Viewing the image we have created * Launching a container using our new image   **Registries**   * What is the Docker hub? * Pushing images to the Docker hub * Running your own internal Docker registry * Testing the internal registry   **A simple use case**   * A single container static website * Setting up a container running Nginx * Launching our static site * Updating our static site from git or bitbucket   **Continuous integration with Docker**   * How Docker enables and supports CI * Getting set up for Jenkins and Docker * A basic Jenkins job * Multi configuration jobs * Drone * Shippable   **A more complex use case: Multi container application stacks**   * A container for our NodeJS application * A base image for our Redis containers * Creating our Redis back-end cluster * Capturing logs * Managing containers   **Integrating with configuration management**   * Managing your Docker hosts with Chef / Puppet / Ansible * Building containers using configuration management tools * Managing running containers with configuration management   [**Ansible**](https://www.simplilearn.com/devops-engineer-masters-program-certification-training?utm_source=google&utm_medium=cpc&utm_term=%2Bdevops%20%2Btraining&utm_content=240527855652&utm_campaign=search-cloud-devops-head-broad-in-Main-Masters-desktop-adgroup-devops-training&gclid=EAIaIQobChMIn4T975_O2QIV0I6PCh2AyQo3EAAYASAAEgJDkfD_BwE)   * What Is Ansible? * Why Ansible? * New Features in Ansible 2.x * Setting up the Learning Environment * Creating an Ansible Home Base * Installing Ansible * Setting up our Test Environment * Target Machine Requirements   **Jumping into Ansible**   * Ansible Configuration Hierarchy * Ansible Hosts Inventory File * Ad-Hoc Commands * Introduction to Playbooks * Playbook Structure * Introduction to Modules * YML basics * Common Modules * Variables and Facts   **Real-World Playbook**   * Real-Life Playbook: Mattermost Deployment * Real-Life Playbook: Walkthroug * Loops, Blocks, Strategies, and Galaxies * Debugging and Troubleshooting * Conditionals and Control Flow * Templating, Control Flow, and Iteration in Jinja2 * Ansible Blocks * Performance and Execution Strategies * Ansible Galaxy * Security   **Ansible in the Cloud**   * Ansible in the Cloud * Ansible AWS Demo * Ansible Tower   **Puppet Configuration Management Boot Camp**  **Automation**   * The importance of automation * Existing automation tools and comparison with Puppet * Why Puppet is a preferred tool * Different Components of Puppet's architecture * Open source Puppet and Puppet Enterprise   **Setting up your development environment**   * Using the Geppetto IDE * Using git to manage your workflow * Using Virtualbox for Vagrant * Using Vagrant to simulate your environment   **The Puppet Master**   * Setting up the Puppet Master * Passenger for scalability * Certificate management * Deploying with git   **Nodes**   * Organizing your nodes * Using an external node classifier * Hiera   **Environments**   * Using environments for different parts of your infrastructure * Environments and Hiera * Dynamic Environments * Git workflow for environments   **The Puppet Forge**   * Using community modules * Using R10K to manage modules   **Custom Module Creation**   * Manifest files * Files and Templates * Custom facts * Using custom facts with Hiera   **Custom Types**   * Parameterized classes * Defined types * Types and providers * Creating a new type   **Reporting**   * Enabling reporting * Store report output * Processing reports   **The Foreman**   * Install The Foreman * Connect The Foreman with Puppet * The Foreman as an external node classifier * Puppet reports in The Foreman   **Exported Resources**   * Exported resource concepts * Installing PuppetDB * Installing and configuring PostgreSQL * Configure PuppetDB to use PostgreSQL * Configure Puppet to use PuppetDB * Declaring exported resources * Collecting exported resources   **Mcollective**   * Installing activemq * Configuring nodes to use activemq * Using mcollective   **Putting it all Together**   * Gather our infrastructure requirements * Import required community Puppet modules * Define our environments * Define our roles * Create custom modules * Define our nodes * Test our infrastructure from beginning to end   **Chef Essentials Workshop - Official Chef Training**  **Chef Resources**   * Using Chef to install packages on your virtual workstation * Using the chef-client command * Creating a basic Chef recipe file * Defining Chef Resources * Test and repair * Lab: The 'file' resource   **Building Cookbooks**   * Modify a recipe * Collaboration and version control * Generating a Chef cookbook * Defining a Chef recipe that sets up a web server * Group exercise: Version Control * Lab: Set up Git * Lab: Setting up web servers   **Chef client**   * Locally applying multiple cookbooks' recipes with chef-client * Applying a run list * Including a recipe from within another recipe     **Testing Cookbooks**   * Using Test Kitchen to verify your recipes converge on a virtual instance * Reading the ServerSpec documentation * Writing and execute tests * Where do tests live? * Group exercise: Test configuration * Lab: Converge the kitchen * Lab: Commit your work   **Details About a System**   * Managing large numbers of servers * Capturing details about a system * Using the node object within a recipe * Using Ruby's string interpolation * Updating the version of a cookbook * Lab: Update the Cookbook version * Lab: Node Details in the Webserver   **Desired State and Data**   * Cleaner recipes * When to use a template resource * Creating a template file * Using ERB tags to display node data in a template * Defininge a template resource * Using kitchen test on the "apache" cookbook * Using chef-clientto apply the "apache" cookbook's "default" recipe * Updating the "apache" cookbook's version for this patch= * Committing the changes   **Local Workstation Installation**   * Installing ChefDK on your laptop * Executing commands to ensure everything is installed * Installing a local editor like Atom * Lab: You will run the following commands and report their versions: $chef, $chef-client, $knife, $ohai, $berks,   $kitchen, $foodcritic, $rubocop  **The Chef Server**   * Connecting to a Chef Server * Managing Additional systems * Managing User traffic * Uploading cookbooks to a Chef Server * Bootstrapping a node * Managing a node via a Chef Server * Hosted Chef   **Managing Multiple Nodes**   * Managing user traffic * Bootstrapping and updating the run\_list * Running chef-client on a node * Appending values to an attribute within a recipe * Versioning cookbooks and uploading to Chef Server * Lab: Another new node * Lab: Test and update the load balancer * Lab: Run $berks install     **Roles**   * Assigning, defining, and configuring * The 'knife' role * Verifying roles * Roles for everyone   **Search**   * Update a Cookbook to Dynamically Use Nodes with the Web Role * Describe the query syntax used in search * Build a search into your recipe code * Create a Ruby Array and Ruby Hash * Update the myhaproxy wrapper cookbook * Updating, load balancing, uploading and running the new search-capable cookbook   **Environments**   * Keeping your infrastructure current * Creating a production environment * Creating an acceptance environment * Deploying a node to an environment * Updating a search query to be more exact * Set new nodes to production   **Sensu - Monitor and Health Check**  **Sensu Components**   * Message Bus * Event Processor * Monitoring Agent * Documented API     **Building Blocks**   * Checks * Handlers * Filters * Mutators     **Hands-on**   * Monitoring workflow * Running Sensu at Scale * Root cause monitoring (Advanced) * Automated remediation (Advanced) * Application & Service Health Monitoring * Metric Collection & Graphing Dashboards * Real-time Analytics * Agent & Agentless monitoring | |  | |