Demo Projects Overview

Description:
List of demo projects that we are building throughout the DevOps Bootcamp.

Purpose:
This should make it easier for you to keep track of building different Git repos and list them in your portfolio, e.g. for your job search.

In the associated Git repositories you can add README files with detailed descriptions of the respective demo project.

Module 5 - Cloud & Infrastructure as Service Basics

Demo Project:
Create server and deploy application on DigitalOcean

Technologies used:
DigitalOcean, Linux, Java, Gradle

Project Description:
- Setup and configure a server on DigitalOcean
- Create and configure a new Linux user on the Droplet (Security best practice)
- Deploy and run a Java Gradle application on Droplet

Module 5: Cloud & Infrastructure as Service Basics

Module 6 - Artifact Repository Manager with Nexus

Demo Project:
Run Nexus on Droplet and Publish Artifact to Nexus

Technologies used:
Nexus, DigitalOcean, Linux, Java, Gradle, Maven

Project Description:
- Install and configure Nexus from scratch on a cloud server
- Create new User on Nexus with relevant permissions
- Java Gradle Project: Build Jar & Upload to Nexus
- Java Maven Project: Build Jar & Upload to Nexus

Module 6: Artifact Repository Manager with Nexus

Module 7 - Containers with Docker

Demo Project:
Use Docker for local development

Technologies used:
Docker, Node.js, MongoDB, MongoExpress

Project Description:
- Create Dockerfile for Nodejs application and build Docker image
- Run Nodejs application in Docker container and connect to MongoDB database container locally.
- Also run MongoExpress container as a UI of the MongoDB database.

Module 7: Containers with Docker

Demo Project:
Docker Compose - Run multiple Docker containers

Technologies used:
Docker, MongoDB, MongoExpress

Project Description:
- Write Docker Compose file to run MongoDB and MongoExpress containers

Module 7: Containers with Docker

Demo Project:
Dockerize Nodejs application and push to private Docker registry

Technologies used:
Docker, Node.js, Amazon ECR

Project Description:
- Write Dockerfile to build a Docker image for a Nodejs application
- Create private Docker registry on AWS (Amazon ECR)
- Push Docker image to this private repository

Module 7: Containers with Docker

Demo Project:
Deploy Docker application on a server with Docker Compose

Technologies used:
Docker, Amazon ECR, Node.js, MongoDB, MongoExpress

Project Description:
- Copy Docker-compose file to remote server
- Login to private Docker registry on remote server to fetch our app image
- Start our application container with MongoDB and MongoExpress services using docker compose

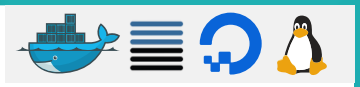
Module 7: Containers with Docker

Demo Project:
Persist data with Docker Volumes

Technologies used:
Docker, Node.js, MongoDB

Project Description:
- Persist data of a MongoDB container by attaching a Docker volume to it

Module 7: Containers with Docker



Demo Project:
Create Docker repository on Nexus and push to it

Technologies used:
Docker, Nexus, DigitalOcean, Linux

Project Description:
- Create Docker hosted repository on Nexus
- Create Docker repository role on Nexus
- Configure Nexus, DigitalOcean Droplet and Docker to be able to push to Docker repository
- Build and Push Docker image to Docker repository on Nexus

Module 7: Containers with Docker

Demo Project:
Deploy Nexus as Docker container

Technologies used:
Docker, Nexus, DigitalOcean, Linux

Project Description:
- Create and Configure Droplet
- Set up and run Nexus as a Docker container

Module 7: Containers with Docker

Module 8 - Build Automation & CI/CD with Jenkins

Demo Project:
Install Jenkins on DigitalOcean

Technologies used:
Jenkins, Docker, DigitalOcean, Linux

Project Description:
- Create an Ubuntu server on DigitalOcean
- Set up and run Jenkins as Docker container
- Initialize Jenkins

Module 8: Build Automation & CI/CD with Jenkins

Demo Project:
Create a CI Pipeline with Jenkinsfile (Freestyle, Pipeline, Multibranch Pipeline)

Technologies used:
Jenkins, Docker, Linux, Git, Java, Maven

Project Description:
CI Pipeline for a Java Maven application to build and push to the repository
- Install Build Tools (Maven, Node) in Jenkins
- Make Docker available on Jenkins server
- Create Jenkins credentials for a git repository
- Create different Jenkins job types (Freestyle, Pipeline, Multibranch pipeline) for the Java Maven project with Jenkinsfile to:
  a. Connect to the application's git repository
  b. Build Jar
  c. Build Docker Image
  d. Push to private DockerHub repository

Module 8: Build Automation & CI/CD with Jenkins

Demo Project:
Create a Jenkins Shared Library

Technologies used:
Jenkins, Groovy, Docker, Git, Java, Maven

Project Description:
Create a Jenkins Shared Library to extract common build logic:
- Create separate Git repository for Jenkins Shared Library project
- Create functions in the JSL to use in the Jenkins pipeline
- Integrate and use the JSL in Jenkins Pipeline (globally and for a specific project in Jenkinsfile)

Module 8: Build Automation & CI/CD with Jenkins

Demo Project:
Configure Webhook to trigger CI Pipeline automatically on every change

Technologies used:
Jenkins, GitLab, Git, Docker, Java, Maven

Project Description:
- Install GitLab Plugin in Jenkins
- Configure GitLab access token and connection to Jenkins in GitLab project settings
- Configure Jenkins to trigger the CI pipeline, whenever a change is pushed to GitLab

Module 8: Build Automation & CI/CD with Jenkins

Demo Project:
Dynamically Increment Application version in Jenkins Pipeline

Technologies used:
Jenkins, Docker, GitLab, Git, Java, Maven

Project Description:
- Configure CI step: Increment patch version
- Configure CI step: Build Java application and clean old artifacts
- Configure CI step: Build Image with dynamic Docker Image Tag
- Configure CI step: Push Image to private DockerHub repository
- Configure CI step: Commit version update of Jenkins back to Git repository
- Configure Jenkins pipeline to not trigger automatically on CI build commit to avoid commit loop

Module 8: Build Automation & CI/CD with Jenkins

Module 9 - AWS Services

Demo Project:
Deploy Web Application on EC2 Instance (manually)

Technologies used:
AWS, Docker, Linux

Project Description:
- Create and configure an EC2 Instance on AWS
- Install Docker on remote EC2 Instance
- Deploy Docker image from private Docker repository on EC2 Instance

Module 9: AWS Services

Demo Project:
CD - Deploy Application from Jenkins Pipeline to EC2 Instance (automatically with docker)

Technologies used:
AWS, Jenkins, Docker, Linux, Git, Java, Maven, Docker Hub

Project Description:
- Prepare AWS EC2 Instance for deployment (Install Docker)
- Create ssh key credentials for EC2 server on Jenkins
- Extend the previous CI pipeline with deploy step to ssh into the remote EC2 instance and deploy newly built image from Jenkins server
- Configure security group on EC2 Instance to allow access to our web application

Module 9: AWS Services

Demo Project:
CD - Deploy Application from Jenkins Pipeline on EC2 Instance (automatically with docker-compose)

Technologies used:
AWS, Jenkins, Docker, Linux, Git, Java, Maven, Docker Hub

Project Description:
- Install Docker Compose on AWS EC2 Instance
- Create docker-compose.yml file that deploys our web application image
- Configure Jenkins pipeline to deploy newly built image using Docker Compose on EC2 server
- Improvement: Extract multiple Linux commands that are executed on remote server into a separate shell script and execute the script from Jenkinsfile

Module 9: AWS Services

Demo Project:
Complete the CI/CD Pipeline (Docker-Compose, Dynamic versioning)

Technologies used:
AWS, Jenkins, Docker, Linux, Git, Java, Maven, Docker Hub

Project Description:
- CI step: Increment version
- CI step: Build artifact for Java Maven application
- CI step: Build and push Docker image to Docker Hub
- CD step: Deploy new application version with Docker Compose
- CD step: Commit the version update

Module 9: AWS Services

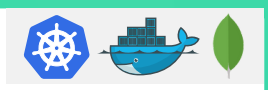
Demo Project:
Interacting with AWS CLI

Technologies used:
AWS, Linux

Project Description:
- Install and configure AWS CLI tool to connect to our AWS account
- Create EC2 Instance using the AWS CLI with all necessary configurations like Security Group
- Create SSH key pair
- Create IAM resources like User, Group, Policy using the AWS CLI
- List and browse AWS resources using the AWS CLI

Module 9: AWS Services

Module 10 - Container Orchestration with Kubernetes

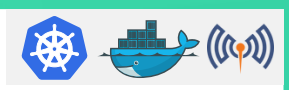


Demo Project:
Deploy MongoDB and Mongo Express into local K8s cluster

Technologies used:
Kubernetes, Docker, MongoDB, Mongo Express

Project Description:
- Setup local K8s cluster with Minikube
- Deploy MongoDB and MongoExpress with configuration and credentials extracted into ConfigMap and Secret

Module 10: Container Orchestration with Kubernetes



Demo Project:
Deploy Mosquitto message broker with ConfigMap and Secret Volume Types

Technologies used:
Kubernetes, Docker, Mosquitto

Project Description:
- Define configuration and passwords for Mosquitto message broker with ConfigMap and Secret Volume types

Module 10: Container Orchestration with Kubernetes

Demo Project:
Install a stateful service (MongoDB) on Kubernetes using Helm

Technologies used:
K8s, Helm, MongoDB, Mongo Express, Linode LKE, Linux

Project Description:
- Create a managed K8s cluster with Linode Kubernetes Engine
- Deploy replicated MongoDB service in LKE cluster using a Helm chart
- Configure data persistence for MongoDB with Linode's cloud storage
- Deploy UI client Mongo Express for MongoDB
- Deploy and configure nginx ingress to access the UI application from browser

Module 10: Container Orchestration with Kubernetes

Demo Project:
Deploy our web application in K8s cluster from private Docker registry

Technologies used:
Kubernetes, Helm, AWS ECR, Docker

Project Description:
- Create Secret for credentials for the private Docker registry
- Configure the Docker registry secret in application Deployment component
- Deploy web application image from our private Docker registry in K8s cluster

Module 10: Container Orchestration with Kubernetes

Demo Project:
Setup Prometheus Monitoring in Kubernetes cluster

Technologies used:
Kubernetes, Helm, Prometheus

Project Description:
- Deploy Prometheus in local Kubernetes cluster using a Helm chart

Module 10: Container Orchestration with Kubernetes

Demo Project:
Deploy Microservices application in Kubernetes with Production & Security Best Practices

Technologies used:
Kubernetes, Redis, Linux, Linode LKE

Project Description:
- Create K8s manifests for Deployments and Services for all microservices of an online shop application
- Deploy microservices to Linode's managed Kubernetes cluster

Module 10: Container Orchestration with Kubernetes

Module 10 - Container Orchestration with Kubernetes

Demo Project:
Create Helm Chart for Microservices

Technologies used:
Kubernetes, Helm

Project Description:
- Create 1 shared Helm Chart for all microservices, to reuse common Deployment and Service configurations for the services

Module 10: Container Orchestration with Kubernetes

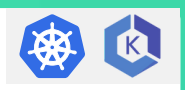
Demo Project:
Deploy Microservices with Helmfile

Technologies used:
Kubernetes, Helm, Helmfile

Project Description:
- Deploy Microservices with Helm
- Deploy Microservices with Helmfile

Module 10: Container Orchestration with Kubernetes

Module 11 - Kubernetes on AWS - EKS

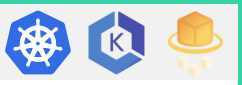


Demo Project:
Create AWS EKS cluster with a Node Group

Technologies used:
Kubernetes, AWS EKS

Project Description:
- Configure necessary IAM Roles
- Create VPC with Cloudformation Template for Worker Nodes
- Create EKS cluster (Control Plane Nodes)
- Create Node Group for Worker Nodes and attach to EKS cluster
- Configure Auto-Scaling of worker nodes
- Deploy a sample application to EKS cluster

Module 11: Kubernetes on AWS - EKS

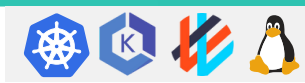


Demo Project:
Create EKS cluster with Fargate profile

Technologies used:
Kubernetes, AWS EKS, AWS Fargate

Project Description:
- Create Fargate IAM Role
- Create Fargate Profile
- Deploy an example application to EKS cluster using Fargate profile

Module 11: Kubernetes on AWS - EKS



Demo Project:
Create EKS cluster with eksctl

Technologies used:
Kubernetes, AWS EKS, Eksctl, Linux

Project Description:
- Create EKS cluster using eksctl tool that reduces the manual effort of creating an EKS cluster

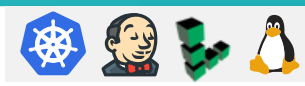
Module 11: Kubernetes on AWS - EKS

Demo Project:
CD - Deploy to EKS cluster from Jenkins Pipeline

Technologies used:
Kubernetes, Jenkins, AWS EKS, Docker, Linux

Project Description:
- Install kubectl and aws-iam-authenticator on a Jenkins server
- Create kubeconfig file to connect to EKS cluster and add it on Jenkins server
- Add AWS credentials on Jenkins for AWS account authentication
- Extend and adjust Jenkinsfile of the previous CI/CD pipeline to configure connection to EKS cluster

Module 11: Kubernetes on AWS - EKS



Demo Project:
CD - Deploy to LKE cluster from Jenkins Pipeline

Technologies used:
Kubernetes, Jenkins, Linode LKE, Docker, Linux

Project Description:
- Create K8s cluster on LKE
- Install kubectl as Jenkins Plugin
- Adjust Jenkinsfile to use Plugin and deploy to LKE cluster

Module 11: Kubernetes on AWS - EKS

Module 11 - Kubernetes on AWS - EKS

Demo Project:
Complete CI/CD Pipeline with EKS and private DockerHub registry

Technologies used:
Kubernetes, Jenkins, AWS EKS, Docker Hub, Java, Maven, Linux, Docker,Git

Project Description:
- Write K8s manifest files for Deployment and Service configuration
- Integrate deploy step in the CI/CD pipeline to deploy newly built application image from DockerHub private registry to the EKS cluster
- So the complete CI/CD project we build has the following configuration:
  a. CI step: Increment version
  b. CI step: Build artifact for Java Maven application
  c. CI step: Build and push Docker image to DockerHub
  d. CD step: Deploy new application version to EKS cluster
  e. CD step: Commit the version update

Module 11: Kubernetes on AWS - EKS

Demo Project:
Complete CI/CD Pipeline with EKS and AWS ECR

Technologies used:
Kubernetes, Jenkins, AWS EKS, AWS ECR, Java, Maven, Linux, Docker, Git

Project Description:
- Create private AWS ECR Docker repository
- Adjust Jenkinsfile to build and push Docker Image to AWS ECR
- Integrate deploying to K8s cluster in the CI/CD pipeline from AWS ECR private registry
- So the complete CI/CD project we build has the following configuration:
  a. CI step: Increment version
  b. CI step: Build artifact for Java Maven application
  c. CI step: Build and push Docker image to AWS ECR
  d. CD step: Deploy new application version to EKS cluster
  e. CD step: Commit the version update

Module 11: Kubernetes on AWS - EKS

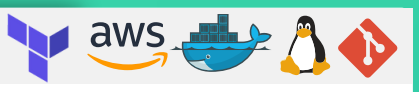
Module 12 - Infrastructure as Code with Terraform

Demo Project:
Automate AWS Infrastructure

Technologies used:
Terraform, AWS, Docker, Linux, Git

Project Description:
- Create TF project to automate provisioning AWS Infrastructure and its components, such as: VPC, Subnet, Route Table, Internet Gateway, EC2, Security Group
- Configure TF script to automate deploying Docker container to EC2 instance

Module 12: Infrastructure as Code with Terraform



Demo Project:
Modularize Project

Technologies used:
Terraform, AWS, Docker, Linux, Git

Project Description:
- Divide Terraform resources into reusable modules

Module 12: Infrastructure as Code with Terraform

Demo Project:
Terraform & AWS EKS

Technologies used:
Terraform, AWS EKS, Docker, Linux, Git

Project Description:
- Automate provisioning EKS cluster with Terraform

Module 12: Infrastructure as Code with Terraform

Demo Project:
Configure a Shared Remote State

Technologies used:
Terraform, AWS S3

Project Description:
- Configure Amazon S3 as remote storage for Terraform state

Module 12: Infrastructure as Code with Terraform

Demo Project:
Complete CI/CD with Terraform

Technologies used:
Terraform, Jenkins, Docker, AWS, Git, Java, Maven, Linux, Docker Hub

Project Description:
Integrate provisioning stage into complete CI/CD Pipeline to automate provisioning server instead of deploying to an existing server
- Create SSH Key Pair
- Install Terraform inside Jenkins container
- Add Terraform configuration to application's git repository
- Adjust Jenkinsfile to add "provision" step to the CI/CD pipeline that provisions EC2 instance
- So the complete CI/CD project we build has the following configuration:
  a. CI step: Build artifact for Java Maven application
  b. CI step: Build and push Docker image to Docker Hub
  c. CD step: Automatically provision EC2 instance using TF
  d. CD step: Deploy new application version on the provisioned EC2 instance with Docker Compose

Module 12: Infrastructure as Code with Terraform

Module 13 - Programming with Python

Demo Project:
Write Countdown Application

Technologies used:
Python, IntelliJ, Git

Project Description:
- Write an application that accepts a user input of a goal and a deadline (date). Print the remaining time until that deadline

Module 13: Programming with Python

Demo Project:
Automation with Python

Technologies used:
Python, IntelliJ, Git

Project Description:
- Write an application that reads a spreadsheet file and process and manipulate the spreadsheet

Module 13: Programming with Python

Demo Project:
API Request to GitLab

Technologies used:
Python, GitLab, IntelliJ, Git

Project Description:
- Write an application that talks to an API of an external application (GitLab) and lists all the public GitLab repositories for a specified user

Module 13: Programming with Python

Module 14 - Automation with Python

Demo Project:
Health Check: EC2 Status Checks

Technologies used:
Python, Boto3, AWS, Terraform

Project Description:
- Create EC2 Instances with Terraform
- Write a Python script that fetches statuses of EC2 Instances and prints to the console
- Extend the Python script to continuously check the status of EC2 Instances in a specific interval

Module 14: Automation with Python

Demo Project:
Automate configuring EC2 Server Instances

Technologies used:
Python, Boto3, AWS

Project Description:
- Write a Python script that automates adding environment tags to all EC2 Server instances

Module 14: Automation with Python

Demo Project:
Automate displaying EKS cluster information

Technologies used:
Python, Boto3, AWS EKS

Project Description:
- Write a Python script that fetches and displays EKS cluster status and information

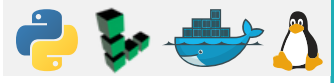
Module 14: Automation with Python

Demo Project:
Data Backup & Restore

Technologies used:
Python, Boto3, AWS

Project Description:
- Write a Python script that automates creating backups for EC2 Volumes
- Write a Python script that cleans up old EC2 Volume snapshots
- Write a Python script that restores EC2 Volumes

Module 14: Automation with Python



Demo Project:
Website Monitoring and Recovery

Technologies used:
Python, Linode, Docker, Linux

Project Description:
- Create a server on a cloud platform
- Install Docker and run a Docker container on the remote server
- Write a Python script that monitors the website by accessing it and validating the HTTP response
- Write a Python script that sends an email notification when website is down
- Write a Python script that automatically restarts the application & server when the application is down

Module 14: Automation with Python

Module 15 - Configuration Management with Ansible

Demo Project:
Automate Node.js application Deployment

Technologies used:
Ansible, Node.js, DigitalOcean, Linux

Project Description:
- Create Server on DigitalOcean
- Write Ansible Playbook that installs necessary technologies, creates Linux user for an application and deploys a NodeJS application with that user

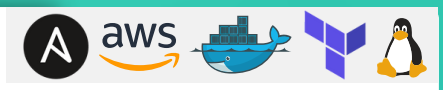
Module 15: Configuration Management with Ansible

Demo Project:
Automate Nexus Deployment

Technologies used:
Ansible, Nexus, DigitalOcean, Java, Linux

Project Description:
- Create Server on DigitalOcean
- Write Ansible Playbook that creates Linux user for Nexus, configure server, installs and deploys Nexus and verifies that it is running successfully

Module 15: Configuration Management with Ansible



Demo Project:
Ansible & Docker

Technologies used:
Ansible, AWS, Docker, Terraform, Linux

Project Description:
- Create AWS EC2 Instance with Terraform
- Write Ansible Playbook that installs necessary technologies like Docker and Docker Compose, copies docker-compose file to the server and starts the Docker containers configured inside the docker-compose file

Module 15: Configuration Management with Ansible

Demo Project:
Ansible Integration in Terraform

Technologies used:
Ansible, Terraform, AWS, Docker, Linux

Project Description:
- Create Ansible Playbook for Terraform integration
- Adjust Terraform configuration to execute Ansible Playbook automatically, so once Terraform provisions a server, it executes an Ansible playbook that configures the server

Module 15: Configuration Management with Ansible

Demo Project:
Configure Dynamic Inventory

Technologies used:
Ansible, Terraform, AWS

Project Description:
- Create EC2 Instance with Terraform
- Write Ansible AWS EC2 Plugin to dynamically sets inventory of EC2 servers that Ansible should manage, instead of hard-coding server addresses in Ansible inventory file

Module 15: Configuration Management with Ansible

Demo Project:
Automate Kubernetes Deployment

Technologies used:
Ansible, Terraform, Kubernetes, AWS EKS, Python, Linux

Project Description:
- Create EKS cluster with Terraform
- Write Ansible Play to deploy application in a new K8s namespace

Module 15: Configuration Management with Ansible

Module 15 - Configuration Management with Ansible

Demo Project:
Ansible Integration in Jenkins

Technologies used:
Ansible, Jenkins, DigitalOcean, AWS, Boto3, Docker, Java, Maven, Linux, Git

Project Description:

- Create and configure a dedicated server for Jenkins
- Create and configure a dedicated server for Ansible Control Node
- Write Ansible Playbook, which configures 2 EC2 Instances
- Add ssh key file credentials in Jenkins for Ansible Control Node server and Ansible Managed Node servers
- Configure Jenkins to execute the Ansible Playbook on remote Ansible Control Node server as part of the CI/CD pipeline
- So the Jenkinsfile configuration will do the following:
  a. Connect to the remote Ansible Control Node server
  b. Copy Ansible playbook and configuration files to the remote Ansible Control Node server
  c. Copy the ssh keys for the Ansible Managed Node servers to the Ansible Control Node server
  d. Install Ansible, Python3 and Boto3 on the Ansible Control Node server
  e. With everything installed and copied to the remote Ansible Control Node server, execute the playbook remotely on that Control Node that will configure the 2 EC2 Managed Nodes

Module 15: Configuration Management with Ansible

Demo Project:
Structure Playbooks with Ansible Roles

Technologies used:
Ansible, Docker, AWS, Linux

Project Description:

- Break up large Ansible Playbooks into smaller manageable files using Ansible Roles

Module 15: Configuration Management with Ansible

Module 16 - Monitoring with Prometheus

Demo Project:
Install Prometheus Stack in Kubernetes

Technologies used:
Prometheus, Kubernetes, Helm, AWS EKS, eksctl, Grafana, Linux

Project Description:
- Setup EKS cluster using eksctl
- Deploy Prometheus, Alert Manager and Grafana in cluster as part of the Prometheus Operator using Helm chart

Module 16: Monitoring with Prometheus

Demo Project:
Configure Alerting for our Application

Technologies used:
Prometheus, Kubernetes, Linux

Project Description:
Configure our Monitoring Stack to notify us whenever CPU usage > 50% or Pod cannot start
- Configure Alert Rules in Prometheus Server
- Configure Alertmanager with Email Receiver

Module 16: Monitoring with Prometheus

Demo Project:
Configure Monitoring for a Third-Party Application

Technologies used:
Prometheus, Kubernetes, Redis, Helm, Grafana

Project Description:
Monitor Redis by using Prometheus Exporter
- Deploy Redis service in our cluster
- Deploy Redis exporter using Helm Chart
- Configure Alert Rules (when Redis is down or has too many connections)
- Import Grafana Dashboard for Redis to visualize monitoring data in Grafana

Module 16: Monitoring with Prometheus

Demo Project:
Configure Monitoring for Own Application

Technologies used:
Prometheus, Kubernetes, Node.js, Grafana, Docker, Docker Hub

Project Description:
- Configure our NodeJS application to collect & expose Metrics with Prometheus Client Library
- Deploy the NodeJS application, which has a metrics endpoint configured, into Kubernetes cluster
- Configure Prometheus to scrape this exposed metrics and visualize it in Grafana Dashboard

Module 16: Monitoring with Prometheus