UDACITY

NANODEGREE PROGRAM SYLLABUS

Cloud Native Application Architecture

Overview

In this program, students will learn to run and manage scalable applications in a cloud native environment, using open source tools and projects like ArgoCD, gRPC and Grafana. Students will learn to identify the best application architecture solutions for an organization's needs, design a microservice architecture by leveraging cloud native tools and patterns, implement best practices in Kubernetes security, and use dashboards to diagnose, troubleshoot and improve site reliability.

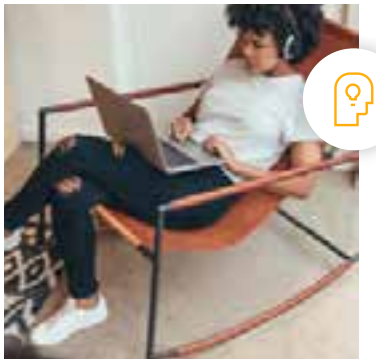
IN COLLABORATION WITH

SUSE

Estimated Time:
4 Months at
10hrs/week

Prerequisites:

- Understand the basics of http
- Basic Python
- Ability to use Git, Linux machines and Linux Command Line
- Familiar with web application development in any language



Flexible Learning:
Self-paced, so you can learn on the schedule that works best for you.

Need Help?
udacity.com/advisor
Discuss this program with an enrollment advisor.

Course 1: Cloud Native Fundamentals

Throughout this course, students will learn how to structure, package and release an application to a Kubernetes cluster, while using an automated CI/CD pipeline. Students will start by applying a suite of good development practices within an application, package it with Docker and distribute it through DockerHub. This will transition to the exploration of Kubernetes resources and how these can be used to deploy an application. At this stage, students will be comfortable using k3s to bootstrap a lightweight and functional Kubernetes cluster. Next, students will examine template configuration managers, such as Helm, to implement the parameterization of Kubernetes declarative manifests. Towards the end of the course, students will learn the fundamentals of Continuous Integration and Continuous Delivery (CI/CD) with GitHub Actions and ArgoCD and completely automate the release process for an application.

Course Project 1:
TechTrends

TechTrends is an online website used as a news sharing platform that enables users to access the latest news within the cloud-native ecosystem. Students will need to extend the project to export and visualize the logs, metrics and status of the application. They will apply their acquired knowledge to package, store and distribute the code as a Docker image. In its turn, the artifact (or Docker image) will be deployed to a cluster using Kubernetes resources, such as deployments and services. By the end of the project, students will use Helm to template the Kubernetes manifests and automate the TechTrends project release using GitHub Actions and ArgoCD.

	LEARNING OUTCOMES	
LESSON ONE	**Welcome to Cloud Native Fundamentals**	• Evaluate the cloud native ecosystem • Explore CNCF (Cloud Native Computing Foundation) and cloud native tooling
LESSON TWO	**Architecture Consideration for Cloud Native Applications**	• Choose monolith or microservice based-architecture for an application • Consider and evaluate the involved trade-offs for monoliths and microservices • Apply good development practices to an application
LESSON THREE	**Container Orchestration with Kubernetes**	• Use Docker to package an application and distribute it via DockerHub • Bootstrap a Kubernetes cluster using k3s • Explore Kubernetes resources for an application deployment • Differentiate between declarative and imperative Kubernetes management techniques

	LEARNING OUTCOMES	
LESSON FOUR	**Open Source PaaS**	• Understand the usage and abstracted components while using a Platform as a Service (PaaS) solution • Explore application deployment with Cloud Foundry
LESSON FIVE	**CI/CD with Cloud Native Tooling**	• Explain CI/CD and its benefits • Apply Continuous Integration fundamentals using GitHub Actions • Apply Continuous Delivery fundamentals using ArgoCD • Use Helm, as a configuration template manager, to parametrize declarative Kubernetes manifests • Deploy an application using ArgoCD and a Helm chart

Course 2: Message Passing

In this course, students will learn how to refactor microservice capabilities from a monolithic architecture and employ different forms of message passing in microservices. To begin, students will create a migration strategy to refactor a service from a monolith to its own microservice and implement the migration. Next, students will be introduced to industry standard best practices for message passing in a service architecture. Finally, students will focus on design decisions and the implementations of different forms of message passing in development and production systems.

Course Project 2:
Refactor UdaConnect

In this project, students will refactor UdaConnect, an existing application that facilitates professional networking at conference and trade shows. UdaConnect ingests and uses location data to find connections between individuals who have been near one another at an event. The current version of the application is built as a proof-of-concept with a monolith architecture. Your task is to apply strategies that you have learned in the course to refactor this application into a microservice architecture and implement message passing strategies to improve its design.

		LEARNING OUTCOMES
LESSON ONE	**Introduction to Message Passing**	• Define message passing • Understand historical context of how and why message passing is used
LESSON TWO	**Refactoring From a Monolith**	• Analyze and identify the first service or capability to decompose a monolith • Create a dependency map in order to prioritize how to refactor a service (based on business logic) • Determine the appropriate migration strategy • Migrate a service from a monolith into its own microservice • Apply the strangler pattern for migrating a monolith architecture

		LEARNING OUTCOMES
LESSON THREE	**Types of Message Passing**	• Identify use cases & implement best practices of REST • Identify use cases of gRPC • Identify use cases & implement best practices of message queues
LESSON FOUR	**Implementing Message Passing**	• Use and apply REST • Use and apply gRPC • Use and apply Kafka
LESSON FIVE	**Message Passing in Production**	• Identify use cases of communication protocol in conjunction with one another • Use OpenAPI • Manage the life cycle of communication protocol

Course 3: Observability

This course covers the fundamentals of observability in distributed systems. Today, Kubernetes has become the de facto standard for cloud native applications and is widely used for distributed systems. To be effective as an observability expert, it is critical to understand how to monitor and respond to the health and performance of both your Kubernetes clusters and the applications hosted on them. This course will teach students how to collect system performance data using Prometheus, how to collect application tracing data using Jaeger and how to visualize the results in a dashboard using Grafana.

Course Project 3: Building a Metrics Dashboard

In this project, you will install and use the basic tools required to perform application tracing and performance monitoring, including Jaeger and Prometheus. You will then learn how to deploy and use Grafana to create dashboards and graphs to visualize performance and trace data collected in the Kubernetes cluster. Finally, you will practice the day-to-day operations of a reliability engineer, such as planning SLIs and filing tickets.

		LEARNING OUTCOMES
LESSON ONE	**Introduction to Cloud Observability**	• Distinguish between black box and white box monitoring • Identify the stakeholders involved in observability • Identify the key tools needed to run a kubernetes cluster
LESSON TWO	**Observability Tools**	• Recognize the distinct roles that Prometheus, Grafana and Jaeger play in observability • Successfully install Prometheus, Grafana and Jaeger on a Kubernetes cluster

		LEARNING OUTCOMES
LESSON THREE	**SLOs, SLIs and Error Budgets**	• Identify the role observability plays in modern applications • Recognize why we use SLOs and SLIs as metrics • Use error budgets to make observability decisions
LESSON FOUR	**Tracing**	• Distinguish tracing from logging and identify the benefits tracing provides beyond standard logging • Identify the basics of how a span is used when tracing applications • Identify the basics of how Jaeger helps manage a trace
LESSON FIVE	**Building Dashboards**	• Navigate Grafana and set up data sources • Create dashboards and panels with various metrics

COURSE 4: Microservices Security

In this course, students will learn how to harden a Docker and Kubernetes microservices architecture. To begin, students will learn STRIDE to threat model and reason about microservice security. Next, students will dig deep to explore the Docker and Kubernetes attack surface and be introduced to industry open-source tools such as Docker-bench and Kube-bench to evaluate and harden Docker and Kubernetes weaknesses. Students will then learn about software composition analysis with Trivy and Grype to evaluate image layers and common application security vulnerabilities and provide remediation. Finally, students will deploy runtime security monitoring to introspect running microservices for security signals and learn how to respond to a security incident.

Course Project 4:
Hardened Microservices Environment

In this project, students will be presented with a real-life scenario to threat-model and harden a Kubernetes environment in response to security concerns brought to them by their company's CTO. Students will use an openSUSE base image to create a hardened Docker container and deploy it to a Docker Hub image registry. Students will then use it to deploy a Kubernetes cluster with a pre-configured Falco DaemonSet and harden the cluster using what we learned from the course. Students will introduce a security incident intentionally, then work on identifying the payload, remediating it and conducting a post-mortem. Students will create alerting for this payload, review lessons learned and write an incident response report.

		LEARNING OUTCOMES
LESSON ONE	**Introduction to Microservices Security**	• Define microservices security • Understand the difference between microservices security and traditional infrastructure security
LESSON TWO	**Threat Modeling with STRIDE**	• Examine the STRIDE methodology for threat modeling as part of the Software Development Lifecycle (SDLC) • Apply the STRIDE methodology to the primary Docker components • Apply the STRIDE methodology to the primary Kubernetes components

	LEARNING OUTCOMES	
LESSON THREE	**Docker Attack Surface Analysis and Hardening**	• Apply Docker security properties in-depth, including client, host and registry, evaluating threat models • Implement CIS benchmarks to harden docker images via docker-bench • Implement image signing using Docker content trust to verify the integrity of the image
LESSON FOUR	**Kubernetes Attack Surface Analysis and Hardening**	• Examine Kubernetes security properties in-depth, including cloud-controller-manager, etcd, kube-apiserver, kube-controller-manager, kube-proxy and kube-scheduler • Evaluate findings against CIS benchmarks and apply a methodology for hardening and testing changes
LESSON FIVE	**Software Composition Analysis**	• Examine examples of supply chain tampering with recent Solarwinds incidents and why software analysis composition is vital to security • Examine common application security vulnerabilities and provide remediation • Examine and remediate common vulnerable libraries and application security vulnerabilities in a Flask application
LESSON SIX	**Runtime Monitoring and Incident Response**	• Examine security considerations for dangerous commands and ongoing runtime security • Implement Sysdig Falco as a DaemonSet with a basic rule set to monitor node processes and send to Grafana for visualization and alerting • Define a security response playbook to triage and respond to alerts

CAPSTONE Project: Uda'CityShop

Uda'CityShop is an e-commerce online shop, where customers can browse the available products and read more details about the available items. In addition, the user will be able to evaluate the costs of products in different currencies and follow the recommendations of browsing products with variate discount rates based on ads. Students will need to extend the project to deploy the application to Kubernetes and observe, analyze and implement changes that would optimize the existing components. Throughout this project, they will use GitHub Actions and ArgoCD to package, build and deploy the application to a Kubernetes cluster. In terms of the observability stack, they will use Grafana dashboards to monitor the resource consumption for each microservice and optimize the Ad service using Python and gRPC. By the end of the project, students will have a refactored Uda'CityShop application, that has an automated CI/CD pipeline, enabled observability and a refactored gRPC service.

Our Classroom Experience

REAL-WORLD PROJECTS

Build your skills through industry-relevant projects. Get personalized feedback from our network of 900+ project reviewers. Our simple interface makes it easy to submit your projects as often as you need and receive unlimited feedback on your work.

KNOWLEDGE

Find answers to your questions with Knowledge, our proprietary wiki. Search questions asked by other students and discover in real-time how to solve the challenges that you encounter.

STUDENT HUB

Leverage the power of community through a simple, yet powerful chat interface built within the classroom. Use Student Hub to connect with your technical mentor and fellow students in your Nanodegree program.

WORKSPACES

See your code in action. Check the output and quality of your code by running them on workspaces that are a part of our classroom.

QUIZZES

Check your understanding of concepts learned in the program by answering simple and auto-graded quizzes. Easily go back to the lessons to brush up on concepts anytime you get an answer wrong.

CUSTOM STUDY PLANS

Work with a mentor to create a custom study plan to suit your personal needs. Use this plan to keep track of your progress toward your goal.

PROGRESS TRACKER

Stay on track to complete your Nanodegree program with useful milestone reminders.

Learn with the Best

Katie Gamanji

ECOSYSTEM ADVOCATE FOR CNCF
(CLOUD NATIVE COMPUTING FOUNDATION)

Katie is the Ecosystem Advocate for CNCF (Cloud Native Computing Foundation), steering the End User Community. Katie's focus is to foster the growth and visibility of the End User Community while bridging the gap with other ecosystem units, such as project maintainers, TOCs and SIGs. In the past roles, Katie contributed to the build-out of platforms that gravitate towards cloud-native principles and open-source tooling, with Kubernetes as the focal point. These projects started with the automation of application delivery on OpenStack-based infrastructure, which transitioned into the creation of a centralized, globally distributed platform at Condé Nast.

Justin Lee

DATA PLATFORM ENGINEER AT
STITCH FIX

Justin is an engineer specializing in designing modern data platforms and scalable systems. He has been a consultant for Fortune 500 companies and has traveled the world to work with his clients. He provides mentorship and interviews developers through Codementor and has a BS in Computer Science from UCLA.

Learn with the Best

Nick Reva

TECHNICAL MANAGER, ENGINEERING SECURITY, SNAPCHAT

Nick has been working in Security Engineering his entire career, for over 14 years now. Over the last five years, he's been a Technical Manager leading teams to build highly scalable security services at tech companies, including SpaceX and Snapchat. In his current role at Snapchat, he lead security engineering teams to build and maintain security services across our cloud native environment. Snapchat are big fans of open source and CNCF projects like Envoy.

Jason "Jay" Smith

APP MODERNIZATION SPECIALIST AT GOOGLE CLOUD

Jason "Jay" Smith is an App Modernization Specialist at Google Cloud. Jay has over 15 years experience in technology and open source solutions. Currently Jay helps Google Cloud customers modernize their application platforms using best practices in cloud native technologies. Prior to joining Google, Jay had worked for other organizations while also leading his own.

All Our Nanodegree Programs Include:

EXPERIENCED PROJECT REVIEWERS

REVIEWER SERVICES

- Personalized feedback & line by line code reviews
- 1600+ Reviewers with a 4.85/5 average rating
- 3 hour average project review turnaround time
- Unlimited submissions and feedback loops
- Practical tips and industry best practices
- Additional suggested resources to improve

TECHNICAL MENTOR SUPPORT

MENTORSHIP SERVICES

- Questions answered quickly by our team of technical mentors
- 1000+ Mentors with a 4.7/5 average rating
- Support for all your technical questions

PERSONAL CAREER SERVICES

CAREER SUPPORT

- Resume support
- Github portfolio review
- LinkedIn profile optimization

Frequently Asked Questions

PROGRAM OVERVIEW

WHY SHOULD I ENROLL?

Cloud native architecture has been described as the future of application development. This program was designed to help you take advantage of the growing need for skilled cloud native architects.

WHAT JOBS WILL THIS PROGRAM PREPARE ME FOR?

The need for a strong cloud native culture in an enterprise organization is greater than ever. The skills you will gain from this Nanodegree program will qualify you for jobs in several industries as countless companies are trying to keep up with digital transformation.

HOW DO I KNOW IF THIS PROGRAM IS RIGHT FOR ME?

The course is for individuals who are looking to advance their careers as cloud native architects. This is an intermediate course that requires fluency in basic python programming as well as a basic understanding of http, the command line and developing web applications.

ENROLLMENT AND ADMISSION

DO I NEED TO APPLY? WHAT ARE THE ADMISSION CRITERIA?

No. This Nanodegree program accepts all applicants regardless of experience and specific background.

WHAT ARE THE PREREQUISITES FOR ENROLLMENT?

A well-prepared learner will meet the following prerequisites:

- Understand the basics of http (like client, server and internet request)
- Basic Python (data types, Functions, REST requests, web development)
- Ability to use Git, Linux machines and Linux Command Line
- Familiar with web application development in any language
- Familiarity with Docker, exposure to a CI/CD pipeline is not required for success in this program but is a helpful prerequisite skill to have.

IF I DO NOT MEET THE REQUIREMENTS TO ENROLL, WHAT SHOULD I DO?

If you believe you need more preparation, here are some additional Udacity programs that can get you up to speed: Intro to Computer Science, Linux Command Line Basics, Intro to Programming and Front End Web Developer.

FAQs Continued

TUITION AND TERM OF PROGRAM

HOW IS THIS NANODEGREE PROGRAM STRUCTURED?

The Cloud Native Application Architecture Nanodegree program is comprised of content and curriculum to support 5 projects. We estimate that students can complete the program in 4 months working 10 hours per week.

Each project will be reviewed by the Udacity reviewer network. Feedback will be provided and if you do not pass the project, you will be asked to resubmit the project until it passes.

HOW LONG IS THIS NANODEGREE PROGRAM?

Access to this Nanodegree program runs for the length of time specified above. If you do not graduate within that time period, you will continue learning with month to month payments. See the **Terms of Use** and **FAQs** for other policies regarding the terms of access to our Nanodegree programs.

CAN I SWITCH MY START DATE? CAN I GET A REFUND?

Please see the Udacity Nanodegree program FAQs for policies on enrollment in our programs.

SOFTWARE AND HARDWARE

WHAT SOFTWARE AND VERSIONS WILL I NEED IN THIS PROGRAM?

There are no software and version requirements to complete this Nanodegree program. All coursework and projects can be completed via Student Workspaces in the Udacity online classroom.