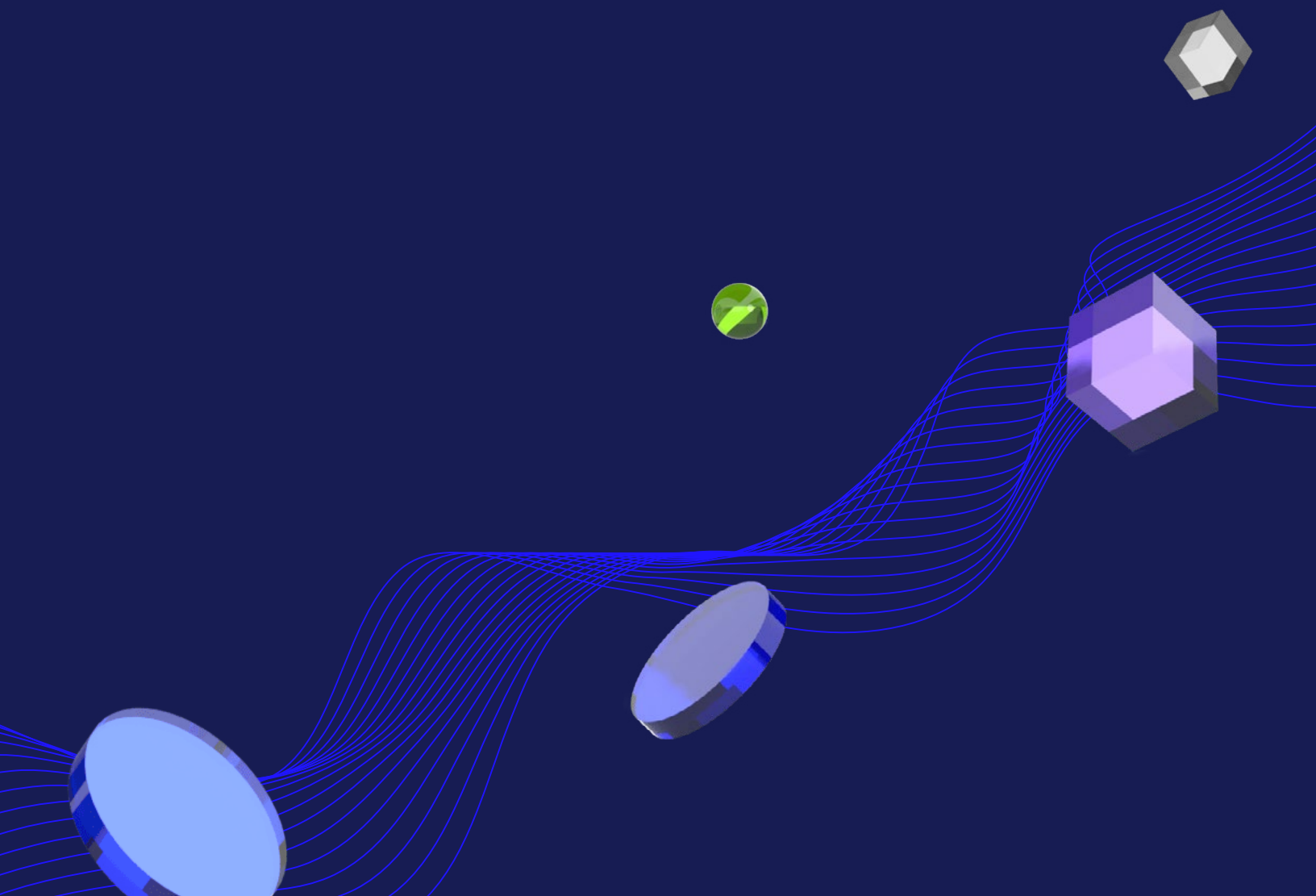
INDIVIDUAL LEARNERS

UDACITY

SCHOOL OF CLOUD COMPUTING

Cloud Native Application Architecture

Nanodegree Program Syllabus



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.

Built in collaboration with:

SUSE

Program information

Estimated Time

4 months at 10hrs/week*

Skill Level

Intermediate

*The length of this program is an estimation of total hours the average student may take to complete all required coursework, including lecture and project time. If you spend about 5-10 hours per week working through the program, you should finish within the time provided. Actual hours may vary.

Prerequisites

A well-prepared learner should:

- Understand the basics of http
- Understand basic Python (data types, functions, REST requests, web development)
- Have the ability to use Git, Linux machines, and Linux command line
- Be familiar with web application development in any language
- Be familiar with Docker (exposure to a CI/CD pipeline can be helpful, but isn't required)

Required Hardware/Software

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.

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. They 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, learners will be comfortable using k3s to bootstrap a lightweight and functional Kubernetes cluster. Next, they 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

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. Learners 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, learners will use Helm to template the Kubernetes manifests, and automate the TechTrends project release using GitHub Actions and ArgoCD.

Lesson 1

Welcome to Cloud Native Fundamentals

- Evaluate the cloud native ecosystem.
- Explore CNCF (Cloud Native Computing Foundation) and cloud native tooling.

Lesson 2

Architecture Consideration for Cloud Native Applications

- Choose monolith or microservice based-architecture for an application.
- Consider and evaluate the involved tradeoffs for monoliths and microservices.
- Apply good development practices to an application.

Lesson 3

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.

Lesson 4

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 5

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, learners will create a migration strategy to refactor a service from a monolith to its own microservice and implement the migration. Next, they will be introduced to industry standard best practices for message passing in a service architecture and will focus on design decisions and the implementations of different forms of message passing in development and production systems.

Course Project

Refactor UdaConnect

In this project, learners 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. The task is to apply strategies from the course to refactor this application into a microservice architecture and implement message passing strategies to improve its design.

Lesson 1

Introduction to Message Passing

- Define message passing.
- Understand historical context of how and why message passing is used.

Lesson 2

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.

Lesson 3

Types of Message Passing

- Identify use cases and implement best practices of REST.
- Identify use cases of gRPC.
- Identify use cases and implement best practices of message queues.

Lesson 4

Implementing Message Passing

- Use and apply REST.
- Use and apply gRPC.
- Use and apply Kafka.

Lesson 5

Message Passing in Production

- Identify use cases of communication protocol in conjunction with one another.
- Use OpenAPI.
- Manage the lifecycle 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 learners how to collect system performance data using Prometheus, collect application tracing data using Jaeger, and visualize the results in a dashboard using Grafana.

Course Project

Building a Metrics Dashboard

In this project, learners will install and use the basic tools required to perform application tracing and performance monitoring, including Jaeger and Prometheus. They 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, learners will practice the day-to-day operations of a reliability engineer, such as planning SLIs and filing tickets.

Lesson 1

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 2

Observability Tools

- Recognize the distinct roles that Prometheus, Grafana, and Jaeger play in observability.
- Successfully install Prometheus, Grafana, and Jaeger on a Kubernetes cluster.

Lesson 3

SLOs, SLIs & 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 4

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 5

Building Dashboards

- Navigate Grafana and set up data sources.
- Create dashboards and panels with various metrics.

Course 4

Microservices Security

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, they 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, they will deploy runtime security monitoring to introspect running microservices for security signals and learn how to respond to a security incident.

Course Project

Hardened Microservices Environment

In this project, learners 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. They will use an openSUSE base image to create a hardened Docker container and deploy it to a Docker Hub image registry. They 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. Learners will introduce a security incident intentionally, then work on identifying the payload, remediating it, and conducting a post-mortem. They will create alerting for this payload, review lessons learned, and write an incident response report.

Lesson 1

Introduction to Microservices Security

- Define microservices security.
- Understand the difference between microservices security and traditional infrastructure security.

Lesson 2

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.

Lesson 3

Docker Attack Surface Analysis & 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 4

Kubernetes Attack Surface Analysis & Hardening

- Examine Kubernetes security properties in-depth, including cloud-controller-manager, etcd, kube-apiserver, kubecontroller-manager, kube-proxy, and kube-scheduler.
- Evaluate findings against CIS benchmarks and apply a methodology for hardening and testing changes.

Lesson 5

Software Composition Analysis

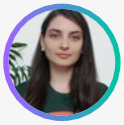
- Examine examples of supply chain tampering with recent Solarwinds incidents and why software analysis composition is vital to security.
- Examine common application security vulnerabilities and provide remediation.
- Examine and remediate common vulnerable libraries and application security vulnerabilities in a Flask application.

Lesson 6

Runtime Monitoring & 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.

Meet your instructors.



Katie Gamanji

Ecosystem Advocate for Cloud Native Computing Foundation

Katie's focus is to foster the growth and visibility of the end user community while bridging the gap with other ecosystem units. In past roles, Katie contributed to the build-out of platforms that gravitate towards cloud-native principles and open-source tooling.

Nick Reva

Technical Manager, Engineering Security at Snapchat

Nick has 14+ years experience in security engineering. He currently leads teams that build highly scalable security services in cloud native environments at companies such as SpaceX and Snapchat.

Justin Lee

Data Platform Engineer at Stitch Fix

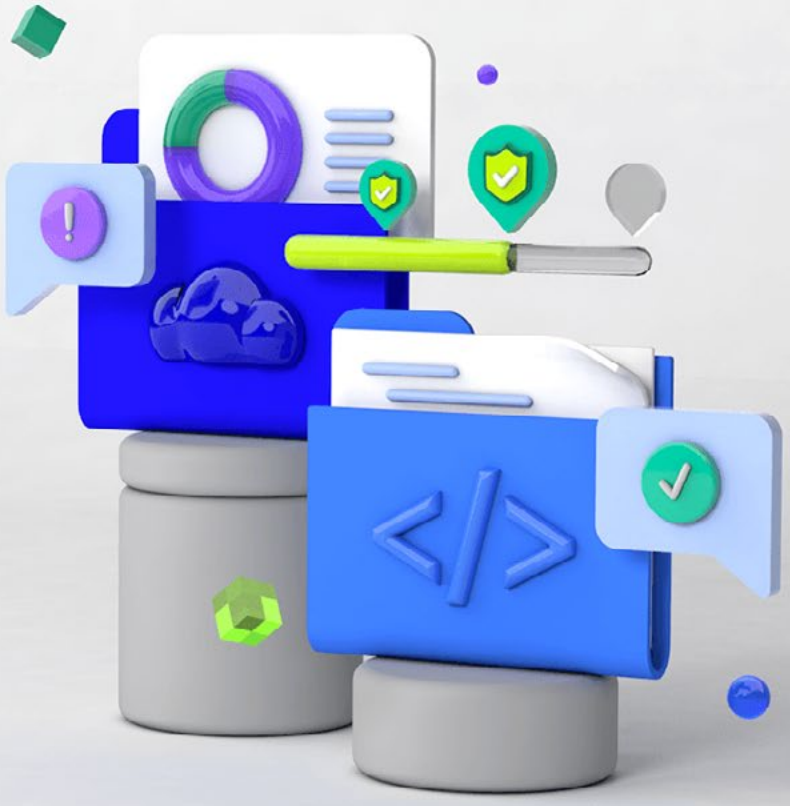
Justin is an engineer specializing in designing modern data platforms and scalable systems. He has been a consultant for Fortune 500 companies and has traveled the world to work with his clients. He provides mentorship through Codementor and has a BS in computer science from UCLA.

Jay Smith

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.

Udacity's learning experience



Hands-on Projects

Open-ended, experiential projects are designed to reflect actual workplace challenges. They aren't just multiple choice questions or step-by-step guides, but instead require critical thinking.

Quizzes

Auto-graded quizzes strengthen comprehension. Learners can return to lessons at any time during the course to refresh concepts.

Knowledge

Find answers to your questions with Knowledge, our proprietary wiki. Search questions asked by other students, connect with technical mentors, and discover how to solve the challenges that you encounter.

Custom Study Plans

Create a personalized study plan that fits your individual needs. Utilize this plan to keep track of movement toward your overall goal.

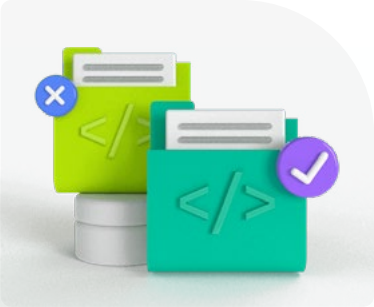
Workspaces

See your code in action. Check the output and quality of your code by running it on interactive workspaces that are integrated into the platform.

Progress Tracker

Take advantage of milestone reminders to stay on schedule and complete your program.

Our proven approach for building job-ready digital skills.



Experienced Project Reviewers

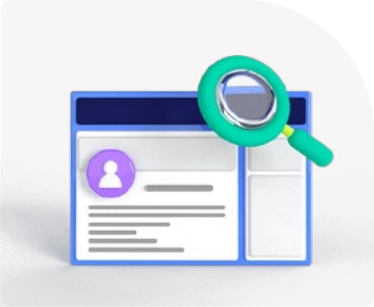
Verify skills mastery.

- Personalized project feedback and critique includes line-by-line code review from skilled practitioners with an average turnaround time of 1.1 hours.
- Project review cycle creates a feedback loop with multiple opportunities for improvement—until the concept is mastered.
- Project reviewers leverage industry best practices and provide pro tips.

Technical Mentor Support

24/7 support unblocks learning.

- Learning accelerates as skilled mentors identify areas of achievement and potential for growth.
- Unlimited access to mentors means help arrives when it's needed most.
- 2 hr or less average question response time assures that skills development stays on track.



Personal Career Services

Empower job-readiness.

- Access to a Github portfolio review that can give you an edge by highlighting your strengths, and demonstrating your value to employers.*
- Get help optimizing your LinkedIn and establishing your personal brand so your profile ranks higher in searches by recruiters and hiring managers.

Mentor Network

Highly vetted for effectiveness.

- Mentors must complete a 5-step hiring process to join Udacity's selective network.
- After passing an objective and situational assessment, mentors must demonstrate communication and behavioral fit for a mentorship role.
- Mentors work across more than 30 different industries and often complete a Nanodegree program themselves.

*Applies to select Nanodegree programs only.

UDACITY

Learn more at

www.udacity.com/online-learning-for-individuals →

01.06.22 | V1.0