Projects Portfolio

VoIP to Cloud Infrastructure Engineer - Career Roadmap

Your Starting Strengths

- Deep SIP/RTP protocol knowledge
- Experience with Avaya/Cisco voice platforms
- Understanding of real-time system requirements
- Troubleshooting complex distributed voice systems
- Quality of Service (QoS) and network optimization

Career Path Options

1. Cloud-Native Telecom Infrastructure Engineer

Companies: Twilio, Vonage API, Bandwidth, SignalWire, RingCentral
Salary Range: $140K-$200K
Focus: Building containerized, scalable voice platforms

2. Site Reliability Engineer (SRE) - Communications

Companies: Zoom, 8x8, Dialpad, Microsoft Teams, Slack
Salary Range: $140K-$220K
Focus: Ensuring reliability and performance of large-scale communication systems

3. Platform Engineering

Companies: Any large tech company, fintech, or enterprise
Salary Range: $130K-$190K
Focus: Building internal platforms that abstract infrastructure complexity

4. Real-Time Systems Engineering

Companies: Trading firms, gaming companies, IoT platforms
Salary Range: $150K-$250K
Focus: Ultra-low latency distributed systems

12-Month Learning Path

Phase 1: Containerization (Months 1-2)

Goal: Master Docker and container concepts

Skills to Learn:

- Docker fundamentals (images, containers, volumes, networks)
- Docker Compose for multi-container applications
- Container networking deep dive
- Health checks and restart policies
- Image optimization and security

Project 1: Containerized SIP Lab

Goal: Build a complete voice environment in Docker

Components:

- Kamailio SIP proxy in Docker

- FreeSWITCH or Asterisk PBX in Docker

- PostgreSQL for subscriber database

- Homer SIP capture for monitoring
- 2-3 softphones for testing

Deliverables:
- Docker Compose file orchestrating all services
- Documentation on GitHub
- Blog post about challenges faced
- Video demo of making calls through your lab

Skills Demonstrated:
- Multi-container orchestration
- Persistent data management
- Network configuration
- Service discovery

Project 2: Auto-Scaling SIP Proxy

Goal: Build monitoring that triggers container scaling

Components:
- Kamailio with Prometheus exporter
- Grafana dashboards showing CPS, active calls
- Script that scales containers based on load
- Load generator to simulate traffic

Deliverables:
- GitHub repo with all code
- Grafana dashboard JSON
- Performance test results showing scaling

Skills Demonstrated:
- Metrics collection
- Automation scripting
- Performance testing
- Threshold-based scaling

Phase 2: Kubernetes (Months 3-4)

Goal: Deploy and manage stateful telecom apps on Kubernetes

Skills to Learn:

- Kubernetes architecture (pods, services, deployments)
- StatefulSets for stateful applications
- ConfigMaps and Secrets management
- Persistent volumes and storage classes
- Service types (ClusterIP, NodePort, LoadBalancer)
- Ingress controllers
- Network policies
- Pod scheduling and affinity rules

Certification Target: CKA (Certified Kubernetes Administrator)

Project 3: Kubernetes Voice Platform

Goal: Deploy production-grade voice infrastructure on K8s

Components:
- Kamailio deployed as StatefulSet (maintain SIP sessions)
- FreeSWITCH/Asterisk for media
- PostgreSQL with StatefulSet
- Redis for session state
- MetalLB or similar for LoadBalancer
- Cert-manager for TLS
- Monitoring stack (Prometheus, Grafana)

Challenges to solve:
- Session persistence across pod restarts
- SIP UDP load balancing
- Media server IP addressing
- Database failover
- TLS certificate management

Deliverables:
- Helm charts for entire stack
- CI/CD pipeline (GitHub Actions)
- Disaster recovery documentation
- Load testing results (1000+ CPS)

Skills Demonstrated:
- Complex stateful app orchestration
- Production-ready configurations

- High availability patterns
- Performance at scale

Project 4: Multi-Cluster Voice Failover

Goal: Active-passive voice infrastructure across two K8s clusters

Components:
- Two Kubernetes clusters (can use k3s locally)
- Global load balancer (GeoDNS simulation)
- Shared PostgreSQL with replication
- Automated failover scripts
- Health monitoring

Deliverables:
- Complete infrastructure-as-code
- Failover testing documentation
- Recovery time objectives (RTO) measurements
- Runbook for disaster scenarios

Skills Demonstrated:
- Multi-cluster management
- Database replication
- Disaster recovery
- Health check design

Phase 3: Observability (Months 5-6)

Goal: Master monitoring, logging, and tracing for distributed systems

Skills to Learn:

- Prometheus metrics design
- PromQL query language
- Grafana dashboard creation
- Alertmanager configuration
- Log aggregation (ELK or Loki)
- Distributed tracing (Jaeger)
- SLI/SLO/SLA definitions
- Error budgets

Project 5: Complete Observability Stack

Goal: Full observability for voice platform

Components:
- Prometheus for metrics collection
- Custom Kamailio exporter (if needed)
- Grafana dashboards:
  - Call success rate (ASR)
  - Calls per second (CPS)
  - Post-dial delay (PDD)
  - Network quality (jitter, packet loss)
  - Resource utilization
- Loki for log aggregation
- Jaeger for tracing SIP calls end-to-end
- AlertManager with PagerDuty integration

Deliverables:
- Complete monitoring stack as code
- 10+ production-ready dashboards
- Alert rules with severity levels
- On-call runbook
- Capacity planning documentation

Skills Demonstrated:
- Metrics design for business outcomes
- Alert fatigue prevention
- Debugging distributed systems
- SRE practices

Project 6: Chaos Engineering for Voice

Goal: Prove infrastructure resilience through controlled failures

Experiments:
- Kill random SIP proxy pods
- Introduce network latency/packet loss
- Fill database connections
- Exhaust CPU/memory
- Simulate complete AZ failure

Deliverables:
- Chaos test suite (LitmusChaos or similar)
- Results showing graceful degradation
- Incident reports and fixes
- Improved architecture documentation

Skills Demonstrated:
- Resilience engineering
- Failure mode analysis
- System hardening
- Blameless postmortems

Phase 4: Infrastructure-as-Code (Months 7-8)

Goal: Automate everything with code

Skills to Learn:

- Terraform (AWS, Azure, GCP providers)
- Terraform modules and workspaces
- State management and backends
- Ansible for configuration management
- GitOps principles (ArgoCD, Flux)
- Secret management (Vault, SOPS)
- CI/CD for infrastructure

Project 7: Multi-Cloud Voice Infrastructure

Goal: Deploy identical voice infrastructure across AWS and Azure

Components:
- Terraform modules for:
  - Network setup (VPC, subnets, security groups)
  - Kubernetes clusters (EKS, AKS)
  - Databases (RDS, Azure Database)
  - Load balancers
  - DNS configuration

- Ansible playbooks for OS-level config
- GitOps workflow for K8s manifests
- Vault for secrets

Deliverables:
- Complete Terraform codebase
- Module documentation
- Cost analysis per cloud provider
- Deployment automation (single command)
- Drift detection and remediation

Skills Demonstrated:
- Cloud platform expertise
- Cost optimization
- Security best practices
- Automated compliance

Project 8: Self-Service Developer Platform

Goal: Internal platform for deploying SIP endpoints

Components:
- REST API (Go or Python)
- Terraform Cloud or Atlantis
- Web UI for developers
- Automated provisioning:
  - SIP accounts
  - DID assignments
  - Routing rules
  - Firewall rules
- RBAC and approval workflows

Deliverables:
- API documentation (OpenAPI/Swagger)
- Example client implementations
- Admin and user guides
- Security audit results

Skills Demonstrated:
- Platform thinking

- API design
- Developer experience focus
- Security and compliance

Phase 5: Programming & Automation (Months 9-10)

Goal: Write production-quality code for infrastructure automation

Skills to Learn:

- Go or Python (choose one, Go preferred for infrastructure)
- REST API design
- Unit and integration testing
- Code review practices
- Git workflows (branching, PRs)
- Database programming
- Kubernetes Operators/Controllers

Project 9: Custom Kubernetes Operator

```
Goal: Create a SIPProxy Custom Resource Definition (CRD)

Functionality:
- Define SIPProxy resources in YAML
- Operator automatically provisions:
  - Kamailio pods
  - Database schemas
  - ConfigMaps with routing rules
  - Monitoring dashboards
  - DNS entries
- Handle updates and deletions
- Self-healing capabilities

Example Usage:
apiVersion: telecom.example.com/v1
kind: SIPProxy
metadata:
  name: production-proxy
```

```
spec:
  replicas: 3
  region: us-west-2
  database: postgres-prod
  routes:
    - pattern: "^1[0-9]{10}$"
      destination: "pstn-gateway"
```

Deliverables:
- Open source operator on GitHub
- Comprehensive documentation
- Helm chart for operator
- Demo video

Skills Demonstrated:
- Kubernetes internals
- Advanced Go programming
- Operator pattern
- Open source contribution

Project 10: Call Analytics Pipeline

Goal: Real-time CDR processing and analytics

Components:
- Kafka for CDR streaming
- Stream processing (Kafka Streams or Flink)
- TimescaleDB for time-series storage
- Real-time dashboards
- ML-based anomaly detection (optional)
- API for querying call data

Features:
- Fraud detection
- Usage trending
- Quality monitoring
- Cost analysis
- Customer behavior patterns

Deliverables:

- Complete data pipeline
- API documentation
- Sample analytics queries
- Performance benchmarks (millions of CDRs)

Skills Demonstrated:
- Event streaming architecture
- Time-series databases
- API development
- Big data processing

Phase 6: Advanced Topics (Months 11-12)

Goal: Specialized skills that set you apart

Skills to Learn:

- Service mesh (Istio, Linkerd)
- eBPF for observability and security
- Multi-region active-active patterns
- Cost optimization strategies
- Security hardening
- Compliance frameworks (SOC2, HIPAA)

Project 11: WebRTC-SIP Gateway Platform

Goal: Production-ready WebRTC gateway with auto-scaling

Components:
- WebRTC signaling server (Node.js/Go)
- Janus or Jitsi for media
- Kamailio for SIP interworking
- TURN servers for NAT traversal
- Auto-scaling based on concurrent calls
- Geographic routing
- Built-in STUN/TURN infrastructure

Features:
- Browser-to-PSTN calls
- JWT authentication

- Recording capabilities
- Quality monitoring
- Load balancing

Deliverables:
- Complete platform on GitHub
- Demo application
- Performance benchmarks
- Architecture documentation
- Blog series on design decisions

Skills Demonstrated:
- WebRTC expertise
- Full-stack platform design
- Real-time media handling
- Modern telephony architecture

Project 12: Service Mesh for Voice

Goal: Implement Istio for voice microservices

Components:
- Voice platform broken into microservices:
  - SIP routing
  - Authentication
  - Billing
  - Analytics
  - Media control
- Istio service mesh
- mTLS between services
- Advanced traffic management:
  - Canary deployments
  - A/B testing
  - Circuit breaking
  - Retry policies
- Observability through mesh

Deliverables:
- Migration guide from monolith
- Performance comparison

- Security improvements documentation
- Traffic management recipes

Skills Demonstrated:
- Microservices architecture
- Service mesh expertise
- Zero-trust security
- Advanced traffic patterns

Portfolio & Personal Brand

GitHub Profile

Must-Have Repositories:

1. Voice platform on Kubernetes (pinned)
2. Terraform modules for telecom infrastructure (pinned)
3. Custom K8s operator for SIP (pinned)
4. Monitoring/observability examples
5. Code samples in Go/Python
6. Contributions to open-source telecom projects

Profile README:

- Brief intro highlighting telecom → cloud transition
- Links to blog posts and projects
- Certifications and skills
- Contact information

Technical Blog

Article Ideas:

1. "Migrating Legacy PBX to Kubernetes: Lessons Learned"
2. "Auto-Scaling SIP Proxies: When CPS Metrics Matter"
3. "Stateful SIP on Kubernetes: The Hard Parts"
4. "Observability for Voice: Beyond Uptime Monitoring"
5. "Building a Multi-Region Voice Platform with Terraform"
6. "WebRTC-SIP Gateway: Architecture and Performance"

7. "Why Telecom Engineers Make Great SREs"
8. "Database Strategies for High-Volume CDR Storage"

Platforms: Medium, Dev.to, or personal blog

LinkedIn Optimization

Headline:
"Platform Engineer | Building Scalable Cloud-Native Voice Infrastructure | Ex-Avaya/Cisco"

Summary:

- Lead with modern skills (Kubernetes, IaC, Go)
- Mention deep telecom expertise as differentiator
- Link to GitHub and blog
- Highlight specific achievements with metrics

Experience Reframing:

- Before: "Maintained Avaya PBX for enterprise"
- After: "Managed high-availability voice infrastructure serving 10K+ users with 99.99% uptime, handling 500K+ daily call sessions"

Certifications Worth Getting

High Priority

- **CKA (Certified Kubernetes Administrator)** - Most valuable cert
- **AWS Solutions Architect Associate** or **Azure Administrator** - Pick your cloud
- **Terraform Associate** - Shows IaC competency

Medium Priority

- **CKS (Certified Kubernetes Security Specialist)** - After CKA
- **Prometheus Certified Associate** - Good for SRE roles
- **AWS/Azure Advanced Networking** - Leverages existing knowledge

Lower Priority (Nice to Have)

- **CKAD (Certified Kubernetes Application Developer)**
- **HashiCorp Vault Associate**
- **Linux Foundation Certified SysAdmin**

Note: Certifications help with resume screening, but projects prove you can actually do the work.

Job Search Strategy

Resume Tips

Skills Section Format:

```
Infrastructure & Cloud:
- Kubernetes (CKA), Docker, Helm, Service Mesh
- AWS/Azure, Terraform, Ansible, GitOps
- Prometheus, Grafana, ELK/Loki, Jaeger

Programming & Automation:
- Go, Python, Bash scripting
- REST APIs, gRPC, Database programming
- CI/CD pipelines (GitHub Actions, Jenkins)

Telecommunications:
- SIP/RTP protocols, Kamailio, FreeSWITCH, Asterisk
- WebRTC, VoIP QoS, Codec optimization
- High-availability voice architectures
```

Achievement Examples:

- "Architected and deployed Kubernetes-based voice platform handling 1M+ daily calls with 99.95% SLA"
- "Reduced infrastructure costs by 40% through right-sizing and auto-scaling implementation"
- "Built internal platform reducing SIP endpoint deployment time from days to minutes"
- "Implemented observability stack that reduced MTTR by 60%"

Target Companies by Stage

Stage 1: Get Your Foot In (0-6 months)

- Cloud-focused MSPs needing telecom expertise
- UCaaS providers (they understand your background)
- Telecom vendors modernizing (lighter competition)

Stage 2: Level Up (6-12 months)

- Mid-size tech companies with voice products
- SaaS companies with real-time features
- Fintech or gaming companies (value low-latency experience)

Stage 3: Dream Jobs (12+ months)

- FAANG communication teams (AWS Chime, Azure Comms, Google Meet)
- Top-tier UCaaS/CPaaS (Twilio, Vonage, Bandwidth)
- High-frequency trading or gaming infrastructure
- Staff/Principal engineer roles

Interview Preparation

System Design Questions You'll Ace:

- Design a scalable SIP proxy infrastructure
- Build a global call routing system
- Create a real-time analytics pipeline
- Design disaster recovery for stateful services
- Multi-region active-active voice platform

Questions to Prepare:

- Container orchestration strategies for stateful apps
- Tradeoffs between various load balancing approaches
- Handling database failover in distributed systems
- Observability design for microservices
- Cost optimization strategies

Your Unique Value Proposition: "Most cloud engineers have never dealt with sub-100ms latency requirements, stateful UDP protocols, or five-nines uptime for real-time services. My telecom background means I bring reliability and performance thinking that's uncommon in cloud-native teams."

Learning Resources

Books

- **"Kubernetes in Action"** - Marko Luksa (comprehensive K8s)
- **"Site Reliability Engineering"** - Google (SRE principles)
- **"Designing Data-Intensive Applications"** - Martin Kleppmann (systems design)
- **"Terraform: Up & Running"** - Yevgeniy Brikman
- **"Learning Go"** - Jon Bodner

Online Courses

- **KodeKloud** - CKA/CKAD preparation (best for K8s)
- **A Cloud Guru / Linux Academy** - Cloud certifications
- **Udemy: Stephane Maarek's courses** - AWS, Kafka
- **Coursera: SRE Specialization** - Google's SRE program

Communities

- **CNCF Slack** - Kubernetes, Prometheus channels
- **r/kubernetes** - Reddit community
- **VoIP Users Conference** - Telecom community still
- **SRE Weekly Newsletter** - Keep up with industry

Practice Environments

- **Kubernetes the Hard Way** - Kelsey Hightower (deep dive)
- **KillerCoda** - Interactive K8s scenarios
- **Home lab** - 3-node K8s cluster (Raspberry Pis or old PCs)
- **Cloud free tiers** - AWS, Azure, GCP all have free options

---

90-Day Quick Start Plan

Month 1: Foundation

Week 1-2:

- Set up home lab environment
- Complete Docker fundamentals
- Start Project 1 (Containerized SIP Lab)

Week 3-4:

- Finish Project 1
- Begin Kubernetes tutorials
- Write first blog post about Docker journey

Month 2: Kubernetes Deep Dive

Week 5-6:

- CKA study (2 hours/day)
- Deploy simple apps on K8s
- Join CNCF Slack

Week 7-8:

- Take CKA exam
- Start Project 3 (K8s Voice Platform)
- Update LinkedIn with new skills

Month 3: First Applications

Week 9-10:

- Complete Project 3
- Apply to 5 infrastructure/platform roles
- Write blog post about K8s learnings

Week 11-12:

- Start Project 5 (Observability)
- Network on LinkedIn
- Begin informational interviews

Goal by Day 90: Have 2-3 solid projects on GitHub, CKA certification, actively interviewing

---

Success Metrics

3-Month Goals

- [ ] CKA certification obtained
- [ ] 3 projects completed and documented
- [ ] GitHub profile with consistent commits
- [ ] 2 blog posts published
- [ ] Applied to 20+ relevant positions
- [ ] 5 informational interviews completed

6-Month Goals

- [ ] 6 projects completed
- [ ] Contributing to open-source telecom projects
- [ ] Active blog with 5+ posts
- [ ] First infrastructure/platform role offer
- [ ] Cloud certification (AWS/Azure)

12-Month Goals

- [ ] Senior Platform/Infrastructure Engineer role
- [ ] Salary increase of 30-50%
- [ ] Technical influence (conference talk, popular blog)
- [ ] All core projects completed
- [ ] Strong professional network in cloud-native space

Red Flags to Avoid

 Don't:

- Collect certifications without building projects
- Apply to jobs without updating resume/LinkedIn
- Stop at tutorials - always build real things
- Ignore programming - you need scripting at minimum
- Stay in pure legacy PBX roles
- Get discouraged by job rejections (it's a numbers game)

Do:

- Build in public (GitHub, blog posts)
- Network actively (comment, share, connect)
- Apply even if you don't meet 100% of requirements
- Focus on projects that show infrastructure at scale
- Learn from production issues and write about them
- Stay consistent (1-2 hours daily beats weekend binges)

Final Thoughts

Your Competitive Advantage:

You're not starting from zero - you have 10+ years of understanding what production reliability actually means. Most cloud engineers have never:

- Debugged why calls are dropping at 2am
- Dealt with stateful UDP protocols
- Maintained 99.99% uptime for real-time services
- Understood what sub-100ms latency requirements mean

These experiences are VALUABLE. The market needs people who can:

- Build modern infrastructure AND understand telecom
- Design systems that handle real-time traffic at scale
- Bridge the gap between legacy voice and cloud-native

You're Not Transitioning Careers - You're Modernizing Your Skillset

Keep your domain expertise, add modern tooling. That combination is rare and valuable.

Start with Project 1 this week. Small daily progress compounds into career transformation.

Quick Reference

Essential Tools to Install:

- Docker Desktop
- kubectl, helm
- Terraform

- VS Code with extensions
- Git
- AWS/Azure CLI
- Prometheus, Grafana

Daily Habits:

- Code/lab work: 1-2 hours
- Reading: 30 minutes (blogs, docs)
- Networking: 15 minutes (LinkedIn, Slack)

Weekly Habits:

- GitHub commits: 5+ days
- Blog post progress: 1 hour
- Applications: 5-10 jobs
- Learning: New concept or tool

Community Engagement:

- Answer questions in forums
- Share your projects
- Comment on relevant content
- Attend virtual meetups

This roadmap is your guide, not a rigid prescription. Adjust based on your pace, interests, and opportunities. The goal is progression, not perfection.

Next Step: Start Project 1 today. Set up Docker and begin your containerized SIP lab.

Good luck! □□

Revision #1
Created 10 December 2025 08:30:38 by Cesar Gzz
Updated 21 May 2026 00:45:22 by Cesar Gzz