PRITHA GUHA

Kolkata, India • +91 9038811289 • prithat398@gmail.com • Open to relocation within EMEA region

PROFESSIONAL SUMMARY

Cloud Architect with 11+ years of experience designing and delivering secure, scalable, multi-cloud infrastructure for enterprise customers and internal platforms. Expert in Kubernetes orchestration, Infrastructure as Code (Terraform, CloudFormation, Helm), cloud security, FinOps, and zero-downtime deployments across AWS and GCP. Proven track record reducing infrastructure costs by 63% while improving performance by 40% and achieving enterprise-grade reliability for production SaaS platforms.

CORE TECHNICAL SKILLS

Cloud Platforms

- **AWS**: EC2, EKS, ECS, Fargate, Lambda, IAM, S3, VPC, Route53, RDS, Aurora, CloudWatch, CloudTrail, ECR, ELB, ALB, KMS, Service Catalog, CloudFormation, Organizations, Config, SCP, SSM, WAF, CloudFront, API Gateway, Certificate Manager, SQS, SNS, MSK (Kafka), AWS Backup, ElastiCache, CodeBuild, CodePipeline
- **GCP**: GKE, Compute Engine, Cloud SQL, Cloud Storage, Cloud DNS, IAM, KMS, VPC, Cloud Monitoring, Cloud Logging, Artifact Registry, Cloud Run, Service Accounts, Billing

Kubernetes & Container Orchestration

- Kubernetes (GKE, EKS), Helm (chart development, templating, hooks), Docker, StatefulSets, Deployments, DaemonSets, Jobs, CronJobs
- Horizontal Pod Autoscaling (HPA), Cluster Autoscaler, Network Policies, RBAC, Service Accounts, Workload Identity
- Multi-cluster management, multi-tenancy, blue-green deployments, zero-downtime upgrades
- Container security hardening, image optimization (Debian, Alpine, Ubuntu), vulnerability scanning

Infrastructure as Code

- Terraform (modules, providers, state management, CDK), AWS CloudFormation (nested stacks, custom Lambda functions, CDK)
- Helm chart development (60+ templates), values management, release automation
- tfsec security scanning, infrastructure compliance validation

Databases & Data Infrastructure

- PostgreSQL (optimization, sharding, logical replication, PgBouncer connection pooling, HBA/SSL authentication)
- AWS RDS, Aurora, Cloud SQL, pgBackRest (backup/restore, PITR)
- Redis, ElastiCache, Kafka/MSK, Redpanda, Elasticsearch

CI/CD & DevOps

- GitHub Actions, AWS CodePipeline, CodeBuild, ArgoCD
- Supply chain security (SBOM, provenance attestations), automated vulnerability scanning
- GitOps workflows, Dependabot automation

Observability & Monitoring

- Prometheus (Operator, remote write, service discovery), Grafana (18+ dashboards), Mimir
- ELK Stack (Elasticsearch, Logstash, Kibana, Fluent Bit)
- AWS CloudWatch, CloudTrail, GCP Cloud Monitoring, Cloud Logging
- Custom alerting (20+ alert policies), incident response, runbooks

Security & Compliance

- Zero-trust networking, Network Policies, Security Groups, IAM frameworks
- KMS encryption (at rest and in transit), TLS/mTLS for inter-service communication
- HashiCorp Vault (HA cluster, auto-unseal, Kubernetes auth, dynamic secrets)
- cert-manager (Let's Encrypt, ACME DNS01, certificate lifecycle)
- AWS Organizations, SCPs (CIS Benchmark standards), centralized logging
- SOC2 compliance, penetration testing, vulnerability management

Programming & Scripting

- Python (AWS SDK, GCP client libraries, automation), Go (Terraform provider development)
- Bash, SQL, YAML/JSON configuration management

PROFESSIONAL EXPERIENCE

Cloud Architect | Turbot | September 2019 - Present

Kubernetes Platform Engineering & Orchestration

Multi-Cluster Kubernetes Architecture & Management (GKE & EKS)

- Architected and managed 6 production-grade Kubernetes clusters (GKE and EKS) across multiple regions supporting 1000+ pods, 50+ namespaces, and 50+ application deployments
- Designed sophisticated multi-node-pool strategy with dedicated pools for:
  - Core service node pools with green/blue pools for zero-downtime upgrades
  - Dedicated workspace node pools for customer workloads
  - Container-in-container workloads (Ubuntu + Sysbox)
  - Monitoring workloads
- Reduced Kubernetes upgrade downtime from 45 minutes to 15 seconds through comprehensive architecture restructuring:
  - Implemented rolling updates with careful node cordon and drain strategies
  - Restructured Redis architecture from single stateful instance to dual deployment (stateful + stateless)
  - Engineered precise upgrade sequencing for dependencies and stateful workloads
  - Configured pod disruption budgets to maintain availability during node updates
  - Implemented health checks and readiness probes for automated pod migration
  - Tested and validated upgrade procedures in staging before production rollout
- Managed cluster upgrades from v1.28 to v1.33 with zero service disruption across production, staging, and development environments
- Forked and modified upstream Helm charts to meet custom requirements on multiple occasions, including customizations for security policies, resource limits, and multi-tenancy isolation
- Configured cluster autoscaler with min/max limits, initial node counts, and optimized zone distribution for high availability
- Implemented Horizontal Pod Autoscaling (HPA) for automatic scaling based on CPU/memory metrics
- Performed vertical autoscaling based on Prometheus metrics data analysis to right-size resource requests and limits
- Deployed AWS EKS clusters with managed and self-managed node groups across multiple availability zones
- Implemented private Kubernetes clusters (GKE and EKS) with private endpoints and bastion-only access for enhanced security

Workload Orchestration & Scaling

- Deployed and managed 50+ application Deployments including:
  - API servers (pipes-api, steampipe-api)
  - Worker processes (steampipe-worker, fleet-worker, tailpipe-worker)

  - Web frontends (pipes-web, steampipe-web)
  - Distributed systems (Temporal components: frontend, history, matching, worker)
  - Infrastructure services (Vault, cert-manager, ingress-nginx, Redis, Redpanda)
- Configured 20+ StatefulSets for stateful workloads:
  - Temporal (with PostgreSQL backend)
  - Vault HA (3-node Raft cluster with auto-unseal)
  - Redis (stateful and stateless variants)
  - Redpanda (Kafka alternative)
  - Elasticsearch
  - Dynamic per-customer workspace database pods
- Implemented Horizontal Pod Autoscaling (HPA) for 9+ critical components with advanced scaling policies:
  - Scale up triggers: CPU > 80% OR Memory > 90% (1 pod every 15 seconds)
  - Scale down triggers: CPU < 50% AND Memory < 50% (50% reduction over 60 seconds)
  - Integrated with Kubernetes Metrics Server and Prometheus for custom metrics
- Designed and deployed 100+ Kubernetes Services (ClusterIP, LoadBalancer, Headless) for service discovery and load balancing
- Managed 100+ ConfigMaps and Secrets for application configuration across all environments
- Implemented CronJobs for scheduled tasks:
  - Kubernetes secrets and certificates backup to Vault (daily)
  - Database log retention cleanup
  - Plugin updates and version synchronization
  - Various maintenance and housekeeping tasks

Network Security & Zero-Trust Architecture

- Implemented comprehensive network policies (30+ policies) for zero-trust networking across all namespaces:
  - Default deny-all policies per namespace with explicit allow rules
  - DNS egress policies for external name resolution
  - Component-specific ingress/egress rules based on least-privilege principle
  - Namespace-to-namespace communication policies for service mesh
  - Pod-to-pod communication controls with label selectors

- Enforced network isolation for:
  - Core services (steampipe-cloud namespace)
  - Customer workspaces (shard namespaces with per-tenant isolation)
  - Fleet workers (steampipe-fleet namespace)
  - Ingress controllers with rate limiting
  - Database access layers (PgBouncer network policies)
- Configured TLS/mTLS for inter-service communication:
  - Redis with TLS transport encryption
  - Terraform Vault with mTLS authentication
  - Kafka/MSK with TLS 1.2+
  - Temporal with mTLS between services
  - PgBouncer with HBA/SSL authentication
- Implemented ACME challenge network policies for Let's Encrypt certificate validation (HTTP-01 and DNS-01)

Multi-Tenancy & Shard Management

- Architected dynamic multi-tenant platform with per-customer namespace and resource isolation supporting 100+ customer shards
- Implemented automated shard lifecycle management (create, update, delete) with Helm templates
- Designed per-shard components:
  - Dedicated ingress controllers with custom hostnames
  - PgBouncer connection pooling instances
  - Workspace StatefulSets with persistent volumes
  - Envoy sidecars for service mesh
  - Plugin creators and initialization jobs
- Configured network policies for shard-to-shard isolation preventing cross-tenant data access
- Implemented dynamic DNS zone creation for customer-specific domains with Cloud DNS integration
- Designed workspace management system with:
  - Dynamic StatefulSet creation per customer workspace
  - Per-workspace resource quotas (CPU, memory, storage)
  - Per-workspace network isolation with Network Policies

    - Dynamic PVC provisioning with StorageClasses for workspace data persistence

RBAC & Service Account Management

- Created and managed 20+ Kubernetes service accounts with least-privilege RBAC policies across all namespaces
- Configured Role and RoleBinding resources for:
    - Application service accounts (read-only access to ConfigMaps/Secrets)
    - Backup jobs (secrets and certificates access with time-bound permissions)
    - Monitoring and observability tools (Prometheus, Grafana)
    - CI/CD pipelines (deployment and rollout permissions)
- Implemented GCP Workload Identity for seamless cloud service integration:
    - Kubernetes service accounts mapped to GCP service accounts
    - Eliminated credential management and secret storage for cloud API access
    - Configured IAM bindings for least-privilege cloud resource access
- Designed AWS IAM Roles for Service Accounts (IRSA) for EKS clusters with OIDC provider integration

Helm Chart Development & Package Management

Comprehensive Helm Chart Engineering

- Developed enterprise-grade Steampipe Helm chart with 60+ templates covering complete application stack:
    - Application deployments (API, workers, web frontends)
    - Infrastructure components (Temporal, Vault, Redis, Redpanda)
    - Shard management (dynamic namespace and resource creation per customer)
    - Network policies, RBAC, ConfigMaps, Secrets
    - HPA configurations with custom metrics
    - CronJobs and Jobs for maintenance tasks
- Created Foundation Helm chart for shared infrastructure baseline:
    - Namespace provisioning with labels and annotations
    - Ingress controllers (nginx, per-shard controllers)
    - cert-manager issuers (Let's Encrypt production and staging)
    - Common network policies (DNS egress, default deny)
- Implemented Helm hooks for orchestrated deployments:
    - Pre-install hooks for database schema initialization

    - Post-install hooks for smoke tests and health checks
    - Pre-upgrade hooks for database migrations with rollback support
    - Post-upgrade hooks for cache invalidation and cleanup
- Managed semantic versioning with Chart.yaml and AppVersion tracking

Feature Flag System & Configuration Management

- Designed comprehensive feature flag system with 20+ flags for:
    - PgBouncer enablement (per environment and per shard)
    - Secrets backup to Vault (automated daily backups)
    - HPA enablement (per component with custom policies)
    - Workspace isolation features (network policies, resource quotas)
    - Flowpipe experimental features (container-in-container, volume mounts)
    - Monitoring and alerting toggles
- Maintained 500+ configuration parameters across cluster-specific values files:
    - Production values (steampipe-cloud.yaml)
    - Staging values (steampipe-cloud-latest.yaml)
    - Development values (vh-steampipe.yaml, testpipe.yaml)
- Organized configuration hierarchy:
    - Default values in values.yaml
    - Environment-specific overrides in cluster values files
    - Secret values managed via Vault integration
    - Resource limits and requests per component
    - Replica counts with min/max for autoscaling
    - Database connection pool configurations
    - TLS/SSL certificate settings
    - Monitoring and observability configurations

Helm Release Management & Automation

- Automated Helm chart packaging and release via GitHub Actions:
    - Chart linting and validation with helm lint and kubeval
    - Semantic version bumping based on commit messages
    - Chart packaging with helm package

  - Release to internal Helm chart repository with index.yaml updates
  - Chart testing with helm test before production rollout
- Implemented GitOps workflow with ArgoCD for automated deployments:
  - Helm chart synchronization from Git repositories
  - Automated rollouts with health checks
  - Rollback capabilities on deployment failures
  - Drift detection and auto-sync policies
- Maintained internal Helm chart repository with 50+ chart versions and change documentation

Terraform Infrastructure as Code

AWS Infrastructure Automation & Cloud Architecture

- Built secure, large-scale AWS infrastructure using Terraform modules and AWS CloudFormation (nested stacks, custom Lambda functions)
- Designed multi-account landing zones with AWS Organizations for environment separation (production, staging, development, security, logging)
- Implemented Service Control Policies (SCPs) based on CIS Benchmark standards:
  - Deny public S3 bucket access
  - Enforce encryption at rest (EBS, RDS, S3)
  - Restrict regions for compliance requirements
  - Prevent security group modifications in production
  - Enforce MFA for privileged operations
- Enforced comprehensive security controls:
  - IAM frameworks with least-privilege access and time-bound permissions using IAM roles and session policies
  - Centralized logging with CloudTrail to S3 with log file validation and MFA delete
  - Encryption at rest using KMS customer-managed keys (CMKs) with automatic rotation
  - Encryption in transit with TLS 1.2+ for all service communication
  - Security groups with explicit allow rules and default deny
  - Kubernetes network policies for pod-to-pod communication
- Provisioned and managed AWS EKS clusters with Terraform:
  - EKS cluster creation with managed node groups and Fargate profiles

  - VPC configuration with public/private subnets across 3 availability zones
  - IAM roles for EKS cluster, node groups, and pod execution
  - Security group rules for cluster communication
  - CloudWatch logging integration for control plane logs
  - OIDC provider setup for IAM Roles for Service Accounts (IRSA)
- Built AWS infrastructure components:
  - EC2 instances with Auto Scaling Groups for dynamic capacity
  - ECS/Fargate services for containerized workloads
  - Lambda functions for serverless automation and event processing
  - RDS and Aurora databases with Multi-AZ deployment
  - S3 buckets with versioning, lifecycle policies, and encryption
  - CloudFront distributions with custom SSL certificates
  - Route53 hosted zones with health checks and failover routing
  - ALB/NLB with target groups and listener rules
  - API Gateway REST and HTTP APIs with Lambda integrations
  - SQS/SNS for asynchronous messaging and pub/sub patterns
  - MSK (Kafka) clusters for event streaming
  - ElastiCache (Redis) for caching and session management
  - SSM Parameter Store and Secrets Manager for secret management
  - AWS Backup for centralized backup management
  - WAF with rate limiting and geo-blocking rules

GCP Infrastructure Provisioning with Terraform

- Leveraged official Terraform modules for GCP infrastructure:
  - terraform-google-modules/kubernetes-engine (v36.1.0) for GKE cluster provisioning
  - terraform-google-modules/bastion-host (v8.0.0) for secure SSH access
  - terraform-google-modules/network for VPC and subnet management
- Managed 200+ Terraform resources across 3+ GCP projects with comprehensive state management:
  - Remote state storage in GCS with state locking
  - Terraform workspaces for environment separation
  - State file backups with versioning enabled

    - Variable management with .tfvars files per environment
    - Output management for cross-module dependencies
- Provisioned GKE clusters with advanced configurations:
    - Release channels (UNSPECIFIED for version control, REGULAR for automated updates)
    - Private clusters with private endpoints for enhanced security
    - Master authorized networks (bastion-only access via IAP)
    - Monitoring and logging integration with Cloud Operations
    - Network and subnet configuration with secondary IP ranges
    - IP ranges for pods (CIDR /16) and services (CIDR /20)
    - Cluster autoscaling with min/max node limits
    - Node pool configurations with machine types and image types
    - Workload Identity integration for GCP service access
- Configured VPC networking:
    - VPC creation with custom subnets per GKE cluster
    - Secondary IP ranges for GKE pod and service CIDR blocks
    - Private networking with RFC 1918 IP ranges
    - Cloud NAT for outbound internet access from private instances
    - VPC peering for multi-region and cross-project connectivity
    - VPC flow logs for network traffic analysis and security monitoring
    - Firewall rules with priority-based ordering
- Provisioned Identity-Aware Proxy (IAP) bastion hosts:
    - Bastion host creation with terraform-google-modules/bastion-host
    - IAP TCP forwarding for SSH access without public IPs
    - Service account configuration with least-privilege IAM roles
    - Startup scripts for automated bastion configuration
    - Bastion member access management via IAM bindings
    - Shielded VM features (secure boot, vTPM, integrity monitoring)

Cloud SQL Database Infrastructure with Terraform

- Provisioned and optimized multiple Cloud SQL PostgreSQL instances:
    - Application database (spcloud): db-custom-4-16384 (4 vCPU, 16GB RAM)

- Temporal database: db-custom-10-16384 (10 vCPU, 16GB RAM)
- Development databases: db-custom-2-7680 (2 vCPU, 7.5GB RAM)
- Blue/green database setup for zero-downtime upgrades

- Configured high-availability and disaster recovery features:
  - REGIONAL availability type with automatic failover across zones
  - Automated backups with 7-day retention and point-in-time recovery (PITR)
  - Transaction log retention (2 days for PITR window)
  - Binary logging for replication and recovery
- Implemented database optimization features:
  - Query Insights enabled for performance monitoring
  - Logical decoding enabled for change data capture (CDC)
  - Custom database flags:
    - max_connections (100-500 based on instance size)
    - shared_buffers (25% of RAM)
    - effective_cache_size (75% of RAM)
    - pg_shadow_select_role for password auditing
  - Backup configuration with automated scheduling
- Configured database security:
  - Private IP addresses with VPC peering (no public IPs)
  - SSL/TLS enforcement for all connections
  - Database user management with least-privilege access
  - Integration with Cloud IAM for authentication
  - Deletion protection enabled for production databases
- Managed PostgreSQL version upgrades (15 → 16 → 17) with automated testing and validation

Cloud Storage & Lifecycle Management

- Created and managed 20+ GCS buckets with Terraform:
  - Vault backup buckets (with versioning and 7-year retention)
  - Workspace backup buckets (per region for pgBackRest PITR)
  - API buckets (for application data and user uploads)
  - DataTank storage buckets (for data processing pipelines)

  - Mimir buckets (for long-term Prometheus metrics storage)
  - Terraform state buckets (with state locking and versioning)
- Implemented comprehensive lifecycle policies:
  - Automatic deletion after retention period (1 day to 7 years)
  - Prefix-based deletion rules (e.g., temporal/workspace/ after 1 day)
  - Transition to Nearline storage after 30 days for cost optimization
  - Transition to Coldline storage after 90 days for archival data
  - Transition to Archive storage after 365 days for compliance data
- Configured bucket security and compliance:
  - Uniform bucket-level access control (no object-level ACLs)
  - Versioning enabled for critical backup buckets
  - Object lifecycle management for automated cleanup
  - Encryption at rest with Google-managed or customer-managed keys (CMEK)
  - Public access prevention with org policy enforcement
  - Bucket retention policies for immutable backups (WORM)

Cloud DNS Management with Terraform

- Managed 10+ Cloud DNS managed zones:
  - API domains (cloud.steampipe.io, pipes.turbot.com)
  - Database domains (db.steampipe.io, db.pipes.turbot.com)
  - Dashboard domains (dashboard.steampipe.io, dashboard.pipes.turbot.com)
  - Flowpipe domains (flowpipe.pipes.turbot.com)
  - Internal service domains (vault.internal, temporal.internal)
- Configured DNS records with Terraform:
  - A records for ingress controllers and load balancers (with static external IPs)
  - CNAME records for service aliases
  - MX records for email routing
  - TXT records for domain verification and SPF/DKIM
  - NS records for subdomain delegation
- Implemented multi-tenant DNS architecture:
  - Dynamic DNS zone creation per customer organization

  - Automated subdomain provisioning for customer workspaces
  - DNS-based traffic routing for blue-green deployments
  - Health checks with automatic failover routing

IAM & Security with Terraform

- Created 15+ GCP service accounts with Terraform:
  - Vault KMS service account (for auto-unseal with cryptographic operations)
  - DNS01 solver service account (for Let's Encrypt ACME DNS challenges)
  - Mimir service account (for Prometheus metrics storage in GCS)
  - Workspace backup service accounts (per region for pgBackRest operations)
  - Artifact Registry push service account (for CI/CD image publishing)
  - Monitoring service account (for Cloud Monitoring metric writes)
  - Logging service account (for Cloud Logging log writes)
- Designed custom IAM roles with least-privilege access:
  - DNS01 Solver Role (dns.resourceRecordSets.create, dns.changes.create)
  - Mimir Storage Role (storage.objects.create, storage.objects.get, storage.objects.list)
  - Workspace Backup Bucket Role (storage.buckets.get, storage.objects.*)
  - Vault KMS Role (cloudkms.cryptoKeyVersions.useToEncrypt, cloudkms.cryptoKeyVersions.useToDecrypt)
- Configured Workload Identity bindings:
  - Kubernetes service accounts mapped to GCP service accounts
  - IAM policy bindings with roles/iam.workloadIdentityUser
  - Namespace-scoped service account annotations
- Managed IAM policies and bindings:
  - Project-level IAM bindings for service accounts
  - Resource-level IAM bindings for GCS buckets, Cloud SQL, KMS keys
  - Conditional IAM bindings with CEL expressions for time-based and attribute-based access
  - IAM audit logging for all service account operations

Cloud KMS & Encryption Management

- Provisioned KMS infrastructure with Terraform:
  - KMS key rings per environment (production, staging, development)

    - Crypto keys for Vault auto-unseal (symmetric encryption)
    - Crypto keys for database encryption (CMEK for Cloud SQL)
    - Crypto keys for GCS bucket encryption (customer-managed keys)
- Configured key management policies:
    - Automatic key rotation (100000s period, ~27.7 hours)
    - Key version management with primary version designation
    - Key destruction prevention with destroy_scheduled_duration
    - IAM bindings for service account access to keys
- Implemented encryption at rest:
    - Cloud SQL instances with CMEK encryption
    - GCS buckets with customer-managed encryption keys
    - Persistent volumes with Google-managed encryption keys
    - Secrets encryption in etcd with KMS provider

Terraform Best Practices & Security

- Implemented infrastructure security scanning:
    - tfsec security scanning for Terraform configurations with CI/CD integration
    - Automated detection of security misconfigurations (public S3 buckets, unencrypted resources)
    - Custom tfsec rules for organization-specific compliance requirements
    - Pre-commit hooks for local validation before pushing
- Managed Terraform modules and code organization:
    - Created reusable Terraform modules for common patterns (VPC, GKE, Cloud SQL)
    - Module versioning with semantic versions and changelogs
    - Module documentation with inputs, outputs, and usage examples
    - Module testing with Terratest for automated validation
- Implemented Terraform workflows:
    - terraform plan in pull requests with automated commenting
    - terraform apply with approval gates in production
    - Terraform state locking to prevent concurrent modifications
    - Terraform drift detection with automated alerts
    - Cost estimation with Infracost integration

AWS CloudFormation Infrastructure

- Built complex infrastructure using AWS CloudFormation:
  - Nested stacks for modular infrastructure components (networking, compute, database)
  - Custom Lambda functions for CloudFormation custom resources (DNS updates, certificate validation)
  - CloudFormation StackSets for multi-account, multi-region deployments
  - CloudFormation Change Sets for safe infrastructure updates with preview
- Implemented CloudFormation best practices:
  - Parameter validation with allowed patterns and values
  - Condition functions for environment-specific resources
  - Stack outputs for cross-stack references
  - Stack policies to prevent accidental deletion of critical resources
  - Rollback triggers for automatic rollback on CloudWatch alarm breaches

Database Performance & Optimization

PostgreSQL Performance Tuning & Optimization

- Improved PostgreSQL throughput by ~40% through comprehensive optimization:
  - Query analysis using pg_stat_statements and EXPLAIN ANALYZE
  - Index optimization (B-tree, GIN, BRIN) based on query patterns
  - Query rewriting to eliminate subqueries and improve join strategies
  - Database sharding across multiple instances for horizontal scaling
  - Table partitioning (range, list, hash) for large tables with 100M+ rows
- Implemented PgBouncer connection pooling:
  - Reduced database connections by 80%+ through connection reuse
  - Configured transaction pooling mode for stateless API connections
  - Configured session pooling mode for stateful workspace connections
  - Separate PgBouncer instances for API (10-50 connections) and Fleet workers (50-100 connections)
  - HBA (host-based authentication) with SSL/TLS enforcement
  - Auth user configuration with MD5 and SCRAM-SHA-256 authentication
  - Connection pool tuning: default_pool_size, min_pool_size, reserve_pool_size
- Tuned PostgreSQL configuration parameters:
  - max_connections (100-500 based on instance size and workload)

  - shared_buffers (25% of RAM for cache)
  - effective_cache_size (75% of RAM for query planner)
  - work_mem (4MB-64MB per operation based on complexity)
  - maintenance_work_mem (256MB-2GB for VACUUM and CREATE INDEX)
  - checkpoint_completion_target (0.9 for smoother checkpoints)
  - random_page_cost (1.1 for SSD storage)
  - effective_io_concurrency (200 for SSD parallelism)
- Implemented database monitoring and alerting:
  - Cloud SQL Query Insights for slow query detection
  - Custom CloudWatch/Cloud Monitoring metrics for connection pool saturation
  - Prometheus postgres_exporter for detailed database metrics
  - Grafana dashboards for real-time database performance visualization
  - Proactive alerts for long-running queries, lock contention, replication lag

High Availability & Disaster Recovery

- Designed high availability architecture:
  - Ensured reliability during traffic spikes with auto-scaling configurations
  - Tuned database connection pool settings (PgBouncer) for connection reuse
  - Enabled horizontal autoscaling for application pods based on CPU/memory metrics
  - Vertically scaled database instances based on CloudWatch/Cloud Monitoring data
  - Cross-zone setup with multi-AZ deployment for 99.95% SLA
  - Automatic failover with health checks and connection retry logic
- Implemented comprehensive backup strategies:
  - pgBackRest for enterprise-grade PostgreSQL backups:
    - Full backups (daily) with incremental backups (hourly)
    - Point-in-time recovery (PITR) with WAL archiving
    - Cross-region backup storage in GCS with lifecycle policies
    - Backup encryption with AES-256
    - Backup verification with automated restore testing
    - Retention policies (7 days for frequent, 30 days for daily, 1 year for monthly)
  - AWS RDS automated backups with 7-day retention and manual snapshots

    - Cloud SQL automated backups with transaction log retention (2 days)
    - Cross-region backup replication for disaster recovery

Zero-Downtime Database Migrations

- Reduced PostgreSQL migration downtime from 8 hours to 50 seconds:
    - Implemented logical replication for live data synchronization
    - Blue-green database setup with read replicas
    - Schema migrations using online DDL tools (pg_repack)
    - Data validation and consistency checks before cutover
    - Automated cutover with DNS/connection string updates
    - Rollback procedures with minimal data loss (< 1 minute RPO)
- Managed PostgreSQL version upgrades (13 → 14 → 15 → 16 → 17):
    - Pre-upgrade testing with pg_upgrade --check
    - Compatibility testing with application test suites
    - Extension compatibility validation (PostGIS, pg_stat_statements)
    - Post-upgrade monitoring for performance regressions
    - Rollback plan with database snapshots before upgrade

FinOps & Cost Optimization

Infrastructure Cost Reduction

- Reduced infrastructure costs by 63% in GKE-hosted applications through comprehensive FinOps strategies:
    - Resource right-sizing based on Prometheus metrics analysis (CPU, memory, disk usage)
    - Identified over-provisioned workloads and reduced CPU/memory requests by 30-60%
    - Committed-use discounts (CUDs) for predictable workloads (3-year commitments for 42% savings)
    - Preemptible VMs and Spot instances for non-critical workloads (up to 80% savings)
    - Cluster autoscaler tuning to minimize idle node capacity
    - Node pool consolidation from 10 pools to 6 pools for better utilization
    - Eliminated waste by decommissioning unused namespaces and zombie resources
- Reduced infrastructure costs by 34% in AWS-hosted applications:
    - Reserved Instances (RIs) for RDS, ElastiCache, and EC2 with 1-year and 3-year terms
    - Savings Plans for flexible compute usage (Lambda, Fargate, EC2) with up to 72% savings

  - Auto Scaling Groups with predictive scaling to minimize over-provisioning
  - S3 Intelligent-Tiering for automatic cost optimization based on access patterns
  - EBS volume optimization (gp3 instead of gp2, io2 instead of io1) for 20% savings
  - Spot Instances for batch processing and CI/CD workloads (up to 90% savings)
  - CloudWatch Logs retention policy optimization (7 days for debug logs, 30 days for audit logs)
  - Eliminated unused EBS snapshots, old AMIs, and unattached EIPs
- Optimized logging and storage costs:
  - Reduced Cloud Logging costs by 45% through log filtering and sampling
  - Implemented log-based metrics to reduce log storage volume
  - Configured log sinks to GCS with lifecycle policies for cost-effective archival
  - Reduced Prometheus storage costs by 50% through metric relabeling and downsampling
  - Eliminated high-cardinality metrics that provided minimal value
  - Implemented Prometheus recording rules to pre-aggregate expensive queries
  - GCS lifecycle policies for automated data tiering (Standard → Nearline → Coldline → Archive)
  - S3 lifecycle policies for automatic transition to Glacier after 90 days
- Optimized database costs:
  - Right-sized Cloud SQL instances based on CPU/memory/disk utilization (reduced tiers by 1-2 sizes)
  - Implemented read replicas for read-heavy workloads to reduce primary instance load
  - Automated database scaling with Cloud SQL Proxy for connection multiplexing
  - RDS instance right-sizing with CloudWatch metrics analysis
  - Automated start/stop schedules for non-production databases (nights and weekends)
- Implemented cost monitoring and governance:
  - Built custom Grafana dashboards for cost tracking per cluster, namespace, and application
  - Configured budget alerts in GCP Billing and AWS Budgets with email notifications
  - Implemented resource tagging strategy for cost allocation and chargeback reporting
  - Monthly FinOps reviews with stakeholders to identify optimization opportunities
  - Created cost allocation reports per team, project, and customer for transparency
  - Implemented showback/chargeback models for multi-tenant infrastructure
- Achieved cost optimization without compromising security, performance, or reliability:
  - Maintained 99.95% uptime SLA while reducing costs

- Improved application performance metrics (latency, throughput) during cost optimization
- Retained all security controls (encryption, network policies, IAM) during right-sizing
- Zero customer-facing incidents during cost optimization initiatives

CI/CD & Supply Chain Security

Automated Deployment Pipelines

- Designed CI/CD build environments using GitHub Actions:
  - Automated Helm chart linting, validation, and packaging
  - Automated Docker image building with multi-stage builds for size optimization
  - Automated Terraform plan and apply with approval gates
  - Parallelized testing with matrix builds for multiple Kubernetes versions
  - Caching strategies for dependencies (Go modules, Python packages, npm)
- Implemented AWS CodePipeline and CodeBuild:
  - Source stage with GitHub integration and webhook triggers
  - Build stage with CodeBuild for compilation and testing
  - Deploy stage with CloudFormation and EKS integration
  - Manual approval gates for production deployments
  - Rollback automation on deployment failures
- Ensured supply chain security from code to deployment:
  - Container image attestations with SBOM (Software Bill of Materials)
  - Provenance attestations for build reproducibility and auditing
  - Image signing with cosign for container image verification
  - Dependency scanning with Dependabot and Snyk
  - Automated vulnerability scanning in AWS ECR and GCP Artifact Registry
  - Image scanning with Trivy and Clair for CVE detection
  - Admission controllers (OPA Gatekeeper, Kyverno) to enforce signed images only
- Implemented GitOps workflows:
  - ArgoCD for automated Kubernetes deployments from Git
  - Automated sync policies with health checks and pruning
  - Progressive delivery with Argo Rollouts (canary, blue-green)
  - Automated rollback on failed health checks or error rate spikes

    - Drift detection and automatic reconciliation

Vulnerability Management & Patch Cadence

- Established structured approval workflows for vulnerability remediation:
    - Weekly triage meetings to prioritize CVEs by severity (Critical, High, Medium, Low)
    - Automated dependency updates with Dependabot for GitHub Actions, Terraform modules, Go modules
    - Patch management cadence:
        - Critical vulnerabilities: 48-hour SLA for patching
        - High vulnerabilities: 7-day SLA for patching
        - Medium vulnerabilities: 30-day SLA for patching
        - Low vulnerabilities: quarterly patching cycle
- Prioritized critical patches across:
    - Databases (PostgreSQL security updates with testing in staging first)
    - Kubernetes (cluster and node pool upgrades with blue-green strategy)
    - Third-party services (Vault, Temporal, Redis, Nginx)
    - Application dependencies (Python packages, Go modules)
- Set up Dependabot alerts with automated pull requests:
    - Configured Dependabot for automated dependency scanning
    - Automated PR creation for security updates
    - CI/CD integration for automated testing before merge
    - Automated merge for low-risk updates (patch versions)

Observability, Monitoring & Incident Response

Comprehensive Monitoring Infrastructure

- Built 18+ Grafana dashboards for end-to-end observability:
    - **Kubernetes cluster monitoring**: node CPU/memory/disk, pod resource usage, cluster events
    - **Application dashboards**: API request rate/latency/errors, worker job processing, web frontend metrics
    - **Temporal dashboards**: workflow execution, task queue lag, worker utilization
    - **Database dashboards**: PostgreSQL connections/queries/locks, PgBouncer pool saturation, replication lag
    - **Infrastructure dashboards**: Redis hit rate, Nginx request rate, Elasticsearch cluster health
    - **Shard-specific dashboards**: per-customer resource usage and performance metrics

  - **Cost dashboards**: resource consumption per namespace and application

- Deployed Prometheus Operator for metrics collection:
  - ServiceMonitor and PodMonitor CRDs for automatic service discovery
  - Remote write configuration to central Mimir instance for long-term storage
  - Prometheus federation for cross-cluster metric aggregation
  - Metric relabeling for cost optimization (dropped high-cardinality labels)
  - Retention policies (local: 7 days, remote: 1 year)
- Configured comprehensive alerting:
  - Created 20+ GCP alert policies:
    - Database CPU > 80%, memory > 90%, disk I/O > 80%, storage > 85%
    - Kubernetes cluster, node, and pod critical events
    - Node CPU and memory capacity thresholds
    - Pod resource limits exceeded (CPU throttling, OOM kills)
    - Service request latency p95 > 1s, p99 > 5s
  - Created Prometheus alert rules:
    - Application health checks failing
    - High error rates (5xx errors > 5% of requests)
    - Resource utilization (CPU, memory, disk) exceeding thresholds
    - Certificate expiration warnings (30 days, 7 days, 24 hours)
    - Backup job failures
  - Alert notification channels: Email, Slack, PagerDuty for on-call rotation
- Implemented logging infrastructure:
  - Deployed Fluent Bit as DaemonSet for log collection from all pods
  - Configured log forwarding to Elasticsearch with index templates
  - Implemented log parsing and enrichment (Kubernetes metadata, pod labels)
  - Log aggregation to ELK Stack (Elasticsearch, Logstash, Kibana)
  - Log retention policies: 7 days in Elasticsearch, 30 days in S3/GCS for audit logs
  - Log-based metrics for cost-effective alerting (reduced log storage by 40%)
- Reduced weekly operational review time from 8 hours to 20 minutes:
  - Automated dashboards with key metrics and SLI/SLO tracking

  - Proactive alerting reduced manual log inspection by 90%
  - Summary reports with week-over-week comparisons
  - Automated anomaly detection with machine learning models

Incident Management & Response

- Led systematic debugging of production incidents:
  - On-call rotation with 15-minute response time SLA
  - Incident severity classification (SEV1, SEV2, SEV3)
  - Incident command structure with defined roles (IC, Tech Lead, Comms Lead)
  - Real-time collaboration using Slack incident channels
- Documented incidents with comprehensive postmortems:
  - Timeline of events with exact timestamps
  - Root cause analysis with 5 Whys methodology
  - Impact assessment (customers affected, downtime duration, data loss)
  - Action items with owners and due dates
  - Preventive measures to avoid recurrence
- Implemented preventive controls and proactive alerts:
  - Added synthetic monitoring for critical user journeys
  - Implemented chaos engineering experiments with controlled failure injection
  - Created runbooks for common incident scenarios (database failover, pod crashes, certificate renewal)
  - Improved monitoring coverage for blind spots identified in incidents
  - Automated remediation for common issues (pod restarts, cache invalidation)
- Created health check scripts and automation:
  - Cert-manager health check scripts for certificate renewal verification
  - Certificate testing with openssl for expiration and validity checks
  - Cluster connectivity tests with kubectl and API endpoint verification
  - Database health checks with connection pooling and query latency tests
  - Automated smoke tests after deployments with critical API endpoint validation

Security, Compliance & Certificate Management

HashiCorp Vault Integration & Secrets Management

- Deployed and managed Vault HA cluster (3 nodes with Raft storage):
  - Raft consensus for leader election and data replication
  - Cross-zone deployment for high availability
  - Automated leader election and failover
- Configured Vault auto-unseal using GCP KMS and AWS KMS:
  - Eliminated manual unseal operations for faster cluster recovery
  - KMS key permissions with IAM policies for least-privilege access
  - Unseal key rotation with zero-downtime key version updates
- Implemented Kubernetes authentication:
  - Kubernetes JWT authentication method for pod-to-Vault communication
  - Service account-based authentication with role-based policies
  - Token TTL and renewal policies for security
- Configured dynamic secrets for databases:
  - PostgreSQL dynamic credentials with time-bound leases
  - Automatic credential rotation every 24 hours
  - Role-based access with least-privilege SQL grants
  - Credential revocation on pod termination
- Automated daily backup of Vault Raft snapshots:
  - CronJob for automated Raft snapshot creation
  - Snapshot upload to GCS with encryption
  - 7-year retention policy for compliance requirements
  - Automated restore testing quarterly for DR validation
- Automated backup of Kubernetes secrets to Vault:
  - CronJob for daily backup of all namespace secrets
  - Secrets encrypted with Vault transit engine before storage
  - Retention policies with automated cleanup of old backups
  - Restoration runbooks for disaster recovery scenarios

Certificate Automation with cert-manager

- Managed cert-manager for TLS certificate automation:
  - Let's Encrypt integration with production and staging issuers

  - ACME HTTP-01 challenge solver for domain validation
  - ACME DNS-01 challenge solver for wildcard certificates with Cloud DNS integration
  - Certificate lifecycle management with automatic renewal (30 days before expiry)
- Implemented certificate version upgrades:
  - Incremental cert-manager upgrades from v1.6.0 to v1.19.1
  - CRD upgrades with kubectl apply and validation
  - Backward compatibility testing before production rollout
  - Rollback procedures with CRD versioning
- Configured certificates for 100+ ingress controllers and services:
  - Ingress controllers (nginx, per-shard controllers)
  - Temporal components with mTLS certificates
  - Vault with TLS certificates for UI and API
  - Redis with TLS transport encryption
  - Redpanda with TLS for Kafka protocol
  - PgBouncer with SSL certificates for database connections
- Configured certificate rotation policies:
  - Automatic rotation for most certificates (90-day Let's Encrypt)
  - "Never" rotation policy for Vault certificates (manual control)
  - Certificate monitoring with expiration alerts (30, 7, 1 day before expiry)
- Automated backup of cert-manager Certificate CRDs:
  - CronJob for daily backup to Vault
  - Restoration procedures for certificate recovery
  - Testing of certificate restoration in staging environment

Cloud Security & Compliance

- Led SOC2 audit preparation and compliance:
  - Documented security controls and policies
  - Demonstrated encryption at rest and in transit
  - Provided access control evidence (IAM, RBAC)
  - Showed audit logging and monitoring capabilities
  - Passed SOC2 Type II audit with zero findings

- Led penetration testing engagements:
  - Coordinated with external security firms for penetration testing
  - Provided test environments and access for controlled testing
  - Remediated vulnerabilities identified in penetration tests
  - Implemented preventive controls to address security gaps
  - Re-testing and validation of fixes with security team
- Implemented comprehensive security controls:
  - Encryption at rest using KMS for all data stores (databases, storage, backups)
  - Encryption in transit with TLS 1.2+ for all service communication
  - Network segmentation with Security Groups (AWS) and Network Policies (Kubernetes)
  - IAM least-privilege access with regular access reviews
  - Multi-factor authentication (MFA) enforcement for privileged accounts
  - Security monitoring with GuardDuty (AWS) and Security Command Center (GCP)
  - Vulnerability scanning with automated remediation workflows
  - Security group rules with explicit allow and default deny
  - Kubernetes Pod Security Standards (restricted, baseline) enforcement

System Development & Technical Leadership

Application Development & Automation

- Developed Terraform provider using Go:
  - Custom resource types for internal platform API
  - CRUD operations with API client integration
  - Schema validation and error handling
  - Provider testing with acceptance tests
  - Documentation and usage examples
- Coded Python services using AWS SDK and GCP client libraries:
  - Automation scripts for infrastructure provisioning
  - Database migration scripts with transaction management
  - Backup and restore automation with validation
  - Monitoring scripts for health checks and metrics collection
  - API clients for internal service integration

- Created comprehensive Bash automation scripts:
  - Cert-manager health check scripts with openssl validation
  - Certificate testing and renewal verification with expiration checks
  - Cluster connectivity tests with kubectl and API endpoint validation
  - Upgrade scripts for cert-manager, Kubernetes, and components with rollback logic
  - Backup and restore procedures with automated testing
  - Environment management scripts (activate, deactivate, switch)
- Engineered AI-powered solutions:
  - LLM-powered error triage solution for automated incident analysis
  - MCP (Model Context Protocol) server implementation for AI assistance
  - Contributed to open-source AI tooling ecosystems

Technical Leadership & Collaboration

- Led infrastructure teams with 5-8 engineers:
  - Sprint planning and backlog prioritization
  - Code reviews for Terraform, Helm, and application code
  - Mentoring on Kubernetes best practices and cloud architecture
  - Technical design reviews for new features and capabilities
- Acted as primary architecture contact for technical stakeholders:
  - Architecture decision records (ADRs) for major infrastructure decisions
  - Design documentation with diagrams (architecture, network, data flow)
  - Technical presentations for engineering teams and leadership
  - On-call escalation point for production incidents
- Collaborated with non-technical stakeholders:
  - Translated technical concepts for product managers and executives
  - Cost-benefit analysis for infrastructure investments
  - Roadmap planning with business priorities and technical feasibility
  - Risk assessments for security and compliance initiatives
- Delivered new capabilities through Agile methodologies:
  - 2-week sprints with sprint planning, daily standups, and retrospectives
  - User story refinement with acceptance criteria

  - Demo sessions for stakeholders with working features
  - Continuous improvement based on retrospective action items
- Optimized vendor collaboration:
  - Managed relationships with cloud providers (AWS, GCP) for support and credits
  - Evaluated and selected third-party tools (monitoring, security, backup)
  - Negotiated contracts and licensing with cost optimization
  - Technical evaluations with POCs and benchmarking

Associate / Backend QA Engineer | Cognizant | June 2014 - September 2019

Backend QA Testing & Quality Assurance

- Led backend QA testing for enterprise ETL (Extract, Transform, Load) systems handling 10TB+ daily data volumes
- Developed comprehensive test strategies covering:
  - Data validation and integrity checks across source and target systems
  - Performance testing with load simulation for 1M+ records/hour
  - API testing for REST and SOAP services with automated test suites
  - Database testing with SQL queries for data accuracy validation
  - Regression testing with automated test execution
- Created automated test frameworks using Python and Selenium:
  - End-to-end testing for data pipelines
  - API testing with requests library and JSON validation
  - Database testing with psycopg2 and SQL assertions
  - Test reporting with HTML reports and email notifications
- Identified and documented 500+ defects with detailed reproduction steps and severity classification
- Collaborated with development teams for defect triage and resolution

AWS Infrastructure Development

- Developed production Python applications for AWS infrastructure automation:
  - AWS SDK (Boto3) for EC2, S3, RDS, Lambda operations
  - Automated provisioning scripts for development and testing environments
  - Infrastructure health checks and monitoring scripts

    - Database backup automation with S3 storage
- Built AWS infrastructure using CLI and Terraform:
    - Provisioned EC2 instances with user data scripts for configuration
    - Created S3 buckets with lifecycle policies for data retention
    - Configured RDS instances with automated backups
    - Set up VPCs with public/private subnets and NAT gateways
    - Implemented IAM roles and policies for least-privilege access
- Automated deployment processes:
    - Created shell scripts for application deployment to EC2
    - Implemented blue-green deployment strategies for zero downtime
    - Configured AWS CodeDeploy for automated application updates
    - Set up CloudWatch alarms for application and infrastructure monitoring

---

EDUCATION

Post Graduate Diploma in Management (Executive) | IMT Ghaziabad
July 2019 - October 2020

- Focus on Business Strategy, Operations Management, Financial Management

B.Sc in Computer Science | St. Xavier's College, Kolkata
July 2011 - July 2014

- Core courses: Data Structures, Algorithms, Database Management Systems, Operating Systems, Computer Networks

---

KEY ACHIEVEMENTS & METRICS

Infrastructure Scale

- Managed 3+ production-grade Kubernetes clusters with 1000+ pods across 50+ namespaces
- Deployed and maintained 50+ application Deployments and 20+ StatefulSets
- Provisioned 200+ Terraform resources across AWS and GCP
- Managed 10+ DNS zones with hundreds of DNS records
- Created 18+ Grafana dashboards and 20+ alert policies

Cost Optimization

- Reduced GKE infrastructure costs by 63% through right-sizing, committed-use discounts, and lifecycle policies
- Reduced AWS infrastructure costs by 34% through Reserved Instances, Savings Plans, and optimization
- Cut Prometheus storage costs by 50% through metric relabeling and downsampling
- Reduced Cloud Logging costs by 45% through log filtering and lifecycle policies
- Achieved cost savings while maintaining 99.95% uptime SLA

Performance Improvements

- Improved PostgreSQL throughput by 40% through query optimization, sharding, and PgBouncer
- Reduced database connections by 80%+ with PgBouncer connection pooling
- Reduced Kubernetes upgrade downtime from 45 minutes to 15 seconds with blue-green deployments
- Reduced PostgreSQL migration downtime from 8 hours to 50 seconds with logical replication
- Cut weekly operational review time from 8 hours to 20 minutes with automated dashboards

Security & Compliance

- Led SOC2 Type II audit with zero findings
- Implemented zero-trust networking with 30+ network policies
- Configured encryption at rest and in transit for all data stores
- Established 48-hour SLA for critical vulnerability patching
- Managed penetration testing with 100% vulnerability remediation

Reliability & Availability

- Achieved 99.95% uptime SLA for production SaaS platform
- Implemented multi-zone deployment across 3 availability zones
- Configured automated failover for databases and stateful services
- Created comprehensive disaster recovery procedures with quarterly testing
- Automated backup processes with PITR capabilities

CERTIFICATIONS & PROFESSIONAL DEVELOPMENT

- AWS Well-Architected Framework implementation experience
- CIS Benchmark standards enforcement and compliance
- SOC2 compliance leadership and audit management

- Kubernetes security hardening and zero-trust networking
- Infrastructure as Code best practices (Terraform, CloudFormation, Helm)
- FinOps cost optimization strategies and implementation
- Disaster recovery planning and business continuity

TECHNICAL CONTRIBUTIONS

Open Source & Community

- Contributed to open-source AI tooling ecosystems
- Implemented MCP (Model Context Protocol) server
- Developed custom Terraform providers
- Created Helm charts and templates for community use

Technical Writing & Documentation

- Authored comprehensive runbooks for incident response
- Created detailed architecture documentation with diagrams
- Wrote internal knowledge base articles for team onboarding
- Documented best practices for Kubernetes security and cost optimization

Innovation & AI Integration

- Engineered LLM-powered error triage solution for automated incident analysis
- Implemented AI-assisted operational workflows
- Explored machine learning for anomaly detection in monitoring

---

ADDITIONAL INFORMATION

- **Location**: Kolkata, India
- **Relocation**: Open to relocation within EMEA region
- **Work Style**: Strong collaboration and communication skills with cross-functional teams
- **Leadership**: Proven ability to lead technical teams and mentor engineers
- **Problem Solving**: Systematic approach to debugging complex distributed systems
- **Continuous Learning**: Passionate about infrastructure automation, observability, and emerging technologies

- **Availability**: Willing to work in on-call rotations for production support