AWS Infrastructure Design for Odoo 17

Scalable, Cost-Optimized, Production-Ready Architecture Using Terraform

For 1,000 – 10,000 Concurrent Users

Publisher: BitHost

sales@bithost.in

www.bithost.in

Document Type: Technical Use-Case Report & Training Guide

Version: 1.0

Last Updated: December 30, 2025

Target Audience: Cloud/DevOps Engineers, Solution Architects, Senior Developers, IT Consultants

Technology Stack: Odoo 17, PostgreSQL 15, AWS, Terraform, S3, EC2, ALB, RDS, ElastiCache Redis

📋 Table of Contents

1. Executive Summary & Business Context

1.1 Overview

This comprehensive document provides a detailed blueprint for deploying **Odoo 17 ERP** on Amazon Web Services (AWS) using Infrastructure as Code (IaC) with **Terraform**. The architecture is designed to support scalable workloads ranging from **1,000 to 10,000 concurrent users** while maintaining optimal cost-efficiency and high availability.

🎯 Key Objectives

- **Scalability:** Seamless growth from 1,000 to 10,000+ users
- **Cost Efficiency:** Minimal infrastructure costs through intelligent resource allocation
- **High Availability:** 99.9% uptime with multi-AZ deployment
- **Performance:** Sub-second response times for 95th percentile requests
- **Security:** Enterprise-grade security with encryption at rest and in transit
- **Automation:** Fully automated infrastructure provisioning via Terraform

1.2 Why Odoo 17?

Odoo 17 represents the latest evolution of the world's most popular open-source ERP system. It provides:

- **Comprehensive Business Suite:** CRM, Sales, Inventory, Manufacturing, Accounting, HR, Projects
- **Modern Technology Stack:** Python 3.10+, PostgreSQL 15+, modern web technologies
- **Extensive Customization:** Modular architecture with 30,000+ community apps
- **Active Development:** Regular updates and security patches
- **Cost-Effective:** No licensing fees, pay only for infrastructure

1.3 Why AWS and Terraform?

The combination of AWS and Terraform provides several strategic advantages:

AWS Benefits

- Global infrastructure with 30+ regions and 90+ availability zones
- Comprehensive service ecosystem (compute, storage, database, networking)
- Enterprise-grade security and compliance certifications
- Pay-as-you-go pricing model with cost optimization tools
- Proven track record with millions of customers worldwide

Terraform Benefits

- **Infrastructure as Code:** Version-controlled, repeatable deployments
- **State Management:** Track infrastructure changes over time
- **Multi-Cloud Support:** Portable infrastructure definitions
- **Modularity:** Reusable components for different environments
- **Community Support:** Extensive module library and documentation

1.4 Document Purpose & Audience

This document serves multiple purposes:

1. **Implementation Guide:** Step-by-step instructions for deploying Odoo 17 on AWS
2. **Training Resource:** Comprehensive material for upskilling DevOps teams
3. **Reference Architecture:** Best practices for enterprise ERP deployments
4. **Decision Support:** Cost-benefit analysis for capacity planning

> 📚 **Training Use Case**
>
> This document is specifically designed for a **2-hour training session** covering:

- Cloud architecture fundamentals for ERP systems
- Terraform best practices and project structure
- AWS service selection and configuration
- Performance tuning and optimization techniques
- Security hardening and compliance requirements
- Operational procedures and troubleshooting

2. Architecture Overview & Design Philosophy

2.1 High-Level Architecture

The proposed architecture implements a **three-tier design pattern** optimized for Odoo 17 workloads:

🏗️ Three-Tier Architecture

Tier 1: Presentation Layer (Edge/Load Balancing)

- **Route 53:** DNS management and health-based routing
- **AWS Certificate Manager:** SSL/TLS certificate provisioning
- **Application Load Balancer:** Layer 7 load balancing with sticky sessions
- **CloudFront (Optional):** CDN for static asset acceleration

Tier 2: Application Layer (Compute)

- **EC2 Auto Scaling Group:** Dynamic capacity adjustment
- **Launch Templates:** Standardized instance configuration
- **ElastiCache Redis:** Session storage and application caching
- **S3:** Centralized filestore for attachments
- **Systems Manager:** Configuration and patch management

Tier 3: Data Layer (Persistence)

- **RDS PostgreSQL:** Primary database with Multi-AZ
- **Read Replicas:** Query offloading for reporting
- **Automated Backups:** Point-in-time recovery capability
- **Parameter Groups:** Optimized database configuration

2.2 Design Principles

2.2.1 High Availability

- **Multi-AZ Deployment:** Resources distributed across 2-3 availability zones
- **No Single Point of Failure:** Redundancy at every layer
- **Automatic Failover:** RDS Multi-AZ with synchronous replication
- **Health Checks:** ALB and ASG health monitoring

2.2.2 Scalability

- **Horizontal Scaling:** Add/remove EC2 instances based on demand
- **Auto Scaling Policies:** CPU and request-based scaling
- **Database Read Scaling:** Multiple read replicas for query distribution
- **Storage Auto-Scaling:** RDS and EBS automatic capacity increases

2.2.3 Security

- **Network Isolation:** VPC with public/private subnet separation
- **Least Privilege Access:** IAM roles with minimal permissions
- **Encryption Everywhere:** At-rest and in-transit encryption
- **Secrets Management:** AWS Secrets Manager for credentials

2.2.4 Cost Optimization

- **Right-Sizing:** Appropriately sized instances for workload
- **Reserved Capacity:** RI/Savings Plans for baseline load
- **Auto Scaling:** Scale down during off-peak hours
- **Storage Tiering:** S3 lifecycle policies for archival

2.3 Network Architecture

VPC Design (10.0.0.0/16)

Subnet Type	CIDR Block	Availability Zone	Purpose
Public Subnet 1	10.0.1.0/24	ap-south-1a	ALB, NAT Gateway
Public Subnet 2	10.0.2.0/24	ap-south-1b	ALB, NAT Gateway
Private Subnet 1 (App)	10.0.11.0/24	ap-south-1a	EC2 Odoo Instances
Private Subnet 2 (App)	10.0.12.0/24	ap-south-1b	EC2 Odoo Instances
Private Subnet 3 (DB)	10.0.21.0/24	ap-south-1a	RDS Primary/Replica
Private Subnet 4 (DB)	10.0.22.0/24	ap-south-1b	RDS Replica

Traffic Flow

1. **User Request:** HTTPS request arrives at Route 53
2. **DNS Resolution:** Route 53 resolves to ALB public IP
3. **Load Balancing:** ALB distributes to healthy EC2 instances
4. **Application Processing:** Odoo worker processes request
5. **Database Query:** EC2 queries RDS PostgreSQL
6. **File Access:** EC2 reads/writes S3 via IAM role

7. **Response:** Response flows back through ALB to user

3. Capacity Planning & User Scaling Strategy

3.1 Understanding Odoo Concurrency

Odoo is a multi-process application where each worker handles concurrent user requests. Proper capacity planning requires understanding the relationship between users, workers, and infrastructure.

🔑 Key Capacity Metrics

- **Total Users:** Total registered user accounts
- **Concurrent Users:** Users actively using the system simultaneously
- **Peak Concurrency:** Maximum simultaneous users (typically 15-25% of total)
- **Workers per Instance:** Odoo processes per EC2 instance
- **Users per Worker:** Concurrent users one worker can handle (typically 4-6)

3.2 Capacity Planning Formula

Worker Calculation

The number of Odoo workers is calculated using the formula:

Total Workers = (CPU Cores × 2) + 1

This includes HTTP workers + 1 cron worker + 1 long-polling worker

Concurrency Assumptions

User Tier	Total Users	Peak Concurrency (20%)	Required Workers (6 users/worker)
Small	1,000	200	~34 workers
Medium	5,000	1,000	~167 workers
Large	10,000	2,000	~334 workers

⚠️ Important Considerations

- Concurrency rates vary significantly by industry and usage patterns
- Heavy transactional workloads reduce capacity to 3-4 users per worker
- PDF generation and reporting require minimum 2 workers per instance
- Auto Scaling should maintain 20-30% buffer capacity for peak loads

3.3 EC2 Instance Sizing

Instance Type Selection

Instance Type	vCPU	RAM	Workers	Concurrent Users	Use Case
t3.medium	2	4 GB	4-6	24-36	Dev/Testing Only
t3.large	2	8 GB	4-6	24-36	Small Production (1K users)
t3.xlarge	4	16 GB	8-10	48-60	Medium Production (5K users)

Instance Type	vCPU	RAM	Workers	Concurrent Users	Use Case
t3.2xlarge	8	32 GB	16-18	96-108	Large Production (10K users)

Recommended Configurations per User Tier

1,000 Users Configuration

- **Instance Type:** t3.large (2 vCPU, 8GB RAM)
- **Instance Count:** 4-6 instances
- **Total Workers:** 24-36 workers
- **Workers per Instance:** 6
- **Auto Scaling:** Min 2, Desired 4, Max 10

5,000 Users Configuration

- **Instance Type:** t3.xlarge (4 vCPU, 16GB RAM)
- **Instance Count:** 10-12 instances
- **Total Workers:** 100-120 workers
- **Workers per Instance:** 10
- **Auto Scaling:** Min 4, Desired 10, Max 25

10,000 Users Configuration

- **Instance Type:** t3.xlarge or t3.2xlarge
- **Instance Count:** 20-25 (xlarge) or 10-15 (2xlarge)
- **Total Workers:** 200-300 workers
- **Workers per Instance:** 10-18
- **Auto Scaling:** Min 6, Desired 20, Max 50

3.4 Database Sizing (RDS PostgreSQL)

Instance Class Selection

User Tier	Primary Instance	Read Replicas	Storage	Connections
1,000 Users	db.t3.large (2 vCPU, 8GB)	1× db.t3.large	100GB - 500GB (gp3)	100
5,000 Users	db.r6g.xlarge (4 vCPU, 32GB)	2× db.r6g.large	200GB - 1TB (gp3)	200
10,000 Users	db.r6g.2xlarge (8 vCPU, 64GB)	3× db.r6g.xlarge	500GB - 2TB (gp3)	300

PostgreSQL Configuration Parameters

For db.r6g.xlarge (32GB RAM)

```
# PostgreSQL Parameters
shared_buffers = 6GB                    # 20% of RAM
effective_cache_size = 20GB             # 65% of RAM
work_mem = 32MB
maintenance_work_mem = 2GB
max_connections = 200
checkpoint_timeout = 30min
checkpoint_completion_target = 0.9
min_wal_size = 2GB
max_wal_size = 8GB
random_page_cost = 1.1                  # SSD optimization
effective_io_concurrency = 200          # NVMe optimization
jit = off                               # Disable for Odoo workloads
```

3.5 Storage Requirements

S3 File Storage Estimation

User Tier	Estimated Storage	Monthly Cost (S3 Standard)	With Intelligent Tiering
1,000 Users	50-200 GB	$1.15 - $4.60	$0.85 - $3.40
5,000 Users	250GB - 1TB	$5.75 - $23.00	$4.25 - $17.00
10,000 Users	500GB - 2TB	$11.50 - $46.00	$8.50 - $34.00

Assumptions: 50-200MB per user annually, document-intensive usage

RDS Storage Strategy

- **Storage Type:** gp3 SSD with 3,000 IOPS baseline
- **Auto-Scaling:** Enabled with 1000GB maximum
- **Backup Storage:** 7-30 days retention included
- **Snapshot Storage:** Manual snapshots in S3

3.6 Network Bandwidth Planning

Data Transfer Estimates

User Tier	Concurrent Users	Avg Request Size	Required Bandwidth
1,000 Users	200	200-500 KB	100-500 Mbps
5,000 Users	1,000	200-500 KB	500-2,500 Mbps
10,000 Users	2,000	200-500 KB	1-5 Gbps

4. AWS Service Components Deep Dive

4.1 Virtual Private Cloud (VPC)

The VPC provides network isolation and security for the entire Odoo deployment.

Terraform Configuration

```
module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "~> 5.0"

  name = "odoo-production-vpc"
  cidr = "10.0.0.0/16"

  azs             = ["ap-south-1a", "ap-south-1b"]
  public_subnets  = ["10.0.1.0/24", "10.0.2.0/24"]
  private_subnets = [
    "10.0.11.0/24",  # App tier AZ-A
    "10.0.12.0/24",  # App tier AZ-B
    "10.0.21.0/24",  # DB tier AZ-A
    "10.0.22.0/24"   # DB tier AZ-B
  ]

  enable_nat_gateway   = true
  single_nat_gateway   = false  # One NAT per AZ for HA
  enable_dns_hostnames = true
  enable_dns_support   = true

  tags = {
    Project     = "Odoo-ERP"
    Environment = "Production"
    ManagedBy   = "Terraform"
    Publisher   = "Bithost"
  }
}
```

VPC Components

- **Internet Gateway:** Provides internet access for public subnets

- **NAT Gateways:** Allow outbound internet for private subnets (2× for HA)
- **Route Tables:** Separate routing for public and private subnets
- **Network ACLs:** Stateless firewall at subnet level

4.2 Application Load Balancer (ALB)

ALB Configuration

```
resource "aws_lb" "odoo" {
  name               = "odoo-production-alb"
  internal           = false
  load_balancer_type = "application"
  security_groups    = [aws_security_group.alb.id]
  subnets            = module.vpc.public_subnets

  enable_deletion_protection       = true
  enable_http2                     = true
  enable_cross_zone_load_balancing = true
  idle_timeout                     = 60

  tags = {
    Name        = "Odoo-ALB"
    Environment = "Production"
  }
}

resource "aws_lb_target_group" "odoo" {
  name     = "odoo-target-group"
  port     = 8069
  protocol = "HTTP"
  vpc_id   = module.vpc.vpc_id

  health_check {
    enabled             = true
    healthy_threshold   = 2
    interval            = 30
    matcher             = "200,302,303"
    path                = "/web/health"
    port                = "traffic-port"
    protocol            = "HTTP"
    timeout             = 5
```

```
    unhealthy_threshold = 3
  }

  stickiness {
    type            = "lb_cookie"
    cookie_duration = 86400  # 24 hours (required for Odoo sessions)
    enabled         = true
  }

  deregistration_delay = 30
}

resource "aws_lb_listener" "https" {
  load_balancer_arn = aws_lb.odoo.arn
  port              = 443
  protocol          = "HTTPS"
  ssl_policy        = "ELBSecurityPolicy-TLS-1-2-2017-01"
  certificate_arn   = aws_acm_certificate.odoo.arn

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.odoo.arn
  }
}

resource "aws_lb_listener" "http_redirect" {
  load_balancer_arn = aws_lb.odoo.arn
  port              = 80
  protocol          = "HTTP"

  default_action {
    type = "redirect"
    redirect {
      port        = "443"
      protocol    = "HTTPS"
      status_code = "HTTP_301"
    }
  }
}
```

💡 Why Sticky Sessions?

Odoo requires sticky sessions (session affinity) because:

- CSRF token validation requires session consistency
- In-memory cache is stored per worker process
- Long-polling connections need stable routing
- File uploads must complete to the same instance

4.3 EC2 Auto Scaling Group

Launch Template

```
resource "aws_launch_template" "odoo" {
  name_prefix   = "odoo-17-"
  image_id      = data.aws_ami.ubuntu_2204.id
  instance_type = var.odoo_instance_type

  iam_instance_profile {
    name = aws_iam_instance_profile.odoo.name
  }

  vpc_security_group_ids = [aws_security_group.odoo_app.id]

  block_device_mappings {
    device_name = "/dev/sda1"
    ebs {
      volume_size           = 50
      volume_type           = "gp3"
      iops                  = 3000
      throughput            = 125
      encrypted             = true
      delete_on_termination = true
    }
  }

  user_data = base64encode(templatefile("${path.module}/user_data.sh", {
    db_secret_arn = aws_secretsmanager_secret.db_credentials.arn
    s3_bucket     = aws_s3_bucket.odoo_filestore.id
    aws_region    = var.aws_region
    redis_host    = aws_elasticache_cluster.odoo_redis.cache_nodes[0].addres
```

```
  }))

  tag_specifications {
    resource_type = "instance"
    tags = {
      Name        = "Odoo-17-Instance"
      Environment = var.environment
      Publisher   = "Bithost"
    }
  }

  metadata_options {
    http_endpoint               = "enabled"
    http_tokens                 = "required"  # IMDSv2 only
    http_put_response_hop_limit = 1
  }
}
```

Auto Scaling Group

```
resource "aws_autoscaling_group" "odoo" {
  name                = "odoo-production-asg"
  vpc_zone_identifier = module.vpc.private_subnets
  target_group_arns   = [aws_lb_target_group.odoo.arn]

  min_size                  = var.min_instances
  max_size                  = var.max_instances
  desired_capacity          = var.desired_instances
  health_check_type         = "ELB"
  health_check_grace_period = 300
  default_cooldown          = 300

  launch_template {
    id      = aws_launch_template.odoo.id
    version = "$Latest"
  }

  tag {
    key                 = "Name"
    value               = "Odoo-ASG-Instance"
    propagate_at_launch = true
  }
```

```
  lifecycle {
    create_before_destroy = true
  }
}
```

Scaling Policies

```
# CPU-based scaling
resource "aws_autoscaling_policy" "cpu_scaling" {
  name                   = "odoo-cpu-scaling"
  autoscaling_group_name = aws_autoscaling_group.odoo.name
  policy_type            = "TargetTrackingScaling"

  target_tracking_configuration {
    predefined_metric_specification {
      predefined_metric_type = "ASGAverageCPUUtilization"
    }
    target_value = 70.0
  }
}

# Request count-based scaling
resource "aws_autoscaling_policy" "request_scaling" {
  name                   = "odoo-request-scaling"
  autoscaling_group_name = aws_autoscaling_group.odoo.name
  policy_type            = "TargetTrackingScaling"

  target_tracking_configuration {
    predefined_metric_specification {
      predefined_metric_type = "ALBRequestCountPerTarget"
      resource_label         = "${aws_lb.odoo.arn_suffix}/${aws_lb_target_gr
    }
    target_value = 1000.0  # Requests per target per minute
  }
}

# Scheduled scaling for business hours
resource "aws_autoscaling_schedule" "scale_up_morning" {
  scheduled_action_name  = "scale-up-morning"
  autoscaling_group_name = aws_autoscaling_group.odoo.name
  recurrence             = "0 7 * * MON-FRI"  # 7 AM IST weekdays
  min_size               = var.min_instances
  max_size               = var.max_instances
```

```
  desired_capacity       = var.desired_instances
}

resource "aws_autoscaling_schedule" "scale_down_night" {
  scheduled_action_name  = "scale-down-night"
  autoscaling_group_name = aws_autoscaling_group.odoo.name
  recurrence             = "0 20 * * *"  # 8 PM IST daily
  min_size               = 2
  max_size               = 10
  desired_capacity       = 2
}
```

4.4 RDS PostgreSQL with Read Replicas

Primary Database Instance

```
resource "aws_db_instance" "odoo_primary" {
  identifier = "odoo-production-db"

  engine               = "postgres"
  engine_version       = "15.4"
  instance_class       = var.db_instance_class
  allocated_storage    = 100
  max_allocated_storage = 1000
  storage_type         = "gp3"
  storage_encrypted    = true
  iops                 = 3000

  db_name  = "odoo_production"
  username = "odoo_admin"
  password = random_password.db_password.result

  multi_az               = true
  db_subnet_group_name   = aws_db_subnet_group.odoo.name
  vpc_security_group_ids = [aws_security_group.rds.id]
  parameter_group_name   = aws_db_parameter_group.odoo_postgres.name

  backup_retention_period      = 7
  backup_window                = "03:00-04:00"
  maintenance_window           = "sun:04:00-sun:05:00"
  enabled_cloudwatch_logs_exports = ["postgresql", "upgrade"]
```

```
  deletion_protection = true
  skip_final_snapshot = false
  final_snapshot_identifier = "odoo-final-${formatdate("YYYY-MM-DD-hhmm", ti

  performance_insights_enabled    = true
  performance_insights_retention_period = 7

  tags = {
    Name        = "Odoo-Primary-Database"
    Environment = "Production"
    Publisher   = "Bithost"
  }
}
```

Read Replicas

```
resource "aws_db_instance" "odoo_replica" {
  count = var.db_read_replica_count

  identifier          = "odoo-prod-replica-${count.index + 1}"
  replicate_source_db = aws_db_instance.odoo_primary.identifier

  instance_class            = var.db_replica_instance_class
  auto_minor_version_upgrade = true
  publicly_accessible       = false

  performance_insights_enabled = true

  tags = {
    Name        = "Odoo-Read-Replica-${count.index + 1}"
    Environment = "Production"
    Publisher   = "Bithost"
  }
}
```

Custom Parameter Group

```
resource "aws_db_parameter_group" "odoo_postgres" {
  name   = "odoo-postgres-15-optimized"
  family = "postgres15"
```

```
  # Memory Configuration
  parameter {
    name  = "shared_buffers"
    value = "6291456"  # 6GB in 8KB pages
  }

  parameter {
    name  = "effective_cache_size"
    value = "20971520"  # 20GB in 8KB pages
  }

  parameter {
    name  = "work_mem"
    value = "32768"  # 32MB in KB
  }

  parameter {
    name  = "maintenance_work_mem"
    value = "2097152"  # 2GB in KB
  }

  # Checkpoint Configuration
  parameter {
    name  = "checkpoint_completion_target"
    value = "0.9"
  }

  parameter {
    name  = "checkpoint_timeout"
    value = "1800"  # 30 minutes
  }

  parameter {
    name  = "max_wal_size"
    value = "8192"  # 8GB in MB
  }

  # Connection Configuration
  parameter {
    name  = "max_connections"
    value = "200"
  }

  # Performance Tuning
  parameter {
```

```
    name  = "random_page_cost"
    value = "1.1"  # SSD optimization
  }

  parameter {
    name  = "effective_io_concurrency"
    value = "200"  # NVMe optimization
  }

  parameter {
    name  = "jit"
    value = "0"  # Disable JIT for Odoo
  }

  # Autovacuum Tuning
  parameter {
    name  = "autovacuum_max_workers"
    value = "4"
  }

  parameter {
    name  = "autovacuum_naptime"
    value = "10"  # 10 seconds
  }
}
```

4.5 S3 File Storage

S3 Bucket Configuration

```
resource "aws_s3_bucket" "odoo_filestore" {
  bucket = "odoo-prod-filestore-${random_id.bucket_suffix.hex}"

  tags = {
    Name        = "Odoo-Filestore"
    Environment = "Production"
    Publisher   = "Bithost"
  }
}

resource "aws_s3_bucket_versioning" "filestore" {
  bucket = aws_s3_bucket.odoo_filestore.id
```

```
  versioning_configuration {
    status = "Enabled"
  }
}

resource "aws_s3_bucket_server_side_encryption_configuration" "filestore" {
  bucket = aws_s3_bucket.odoo_filestore.id

  rule {
    apply_server_side_encryption_by_default {
      sse_algorithm = "AES256"
    }
    bucket_key_enabled = true
  }
}

resource "aws_s3_bucket_lifecycle_configuration" "filestore" {
  bucket = aws_s3_bucket.odoo_filestore.id

  rule {
    id     = "intelligent-tiering"
    status = "Enabled"

    transition {
      days          = 30
      storage_class = "INTELLIGENT_TIERING"
    }
  }

  rule {
    id     = "archive-old-versions"
    status = "Enabled"

    noncurrent_version_transition {
      noncurrent_days = 90
      storage_class   = "GLACIER_INSTANT_RETRIEVAL"
    }

    noncurrent_version_expiration {
      noncurrent_days = 365
    }
  }
}
```

```
resource "aws_s3_bucket_public_access_block" "filestore" {
  bucket = aws_s3_bucket.odoo_filestore.id

  block_public_acls       = true
  block_public_policy     = true
  ignore_public_acls      = true
  restrict_public_buckets = true
}
```

IAM Policy for EC2 S3 Access

```
resource "aws_iam_role_policy" "odoo_s3_access" {
  name = "odoo-s3-filestore-access"
  role = aws_iam_role.odoo_instance.id

  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Effect = "Allow"
        Action = [
          "s3:GetObject",
          "s3:PutObject",
          "s3:DeleteObject",
          "s3:ListBucket"
        ]
        Resource = [
          aws_s3_bucket.odoo_filestore.arn,
          "${aws_s3_bucket.odoo_filestore.arn}/*"
        ]
      }
    ]
  })
}
```

4.6 ElastiCache Redis

```
resource "aws_elasticache_subnet_group" "odoo" {
  name       = "odoo-redis-subnet-group"
  subnet_ids = module.vpc.private_subnets
```

```
}

resource "aws_elasticache_cluster" "odoo_redis" {
  cluster_id           = "odoo-redis-cache"
  engine               = "redis"
  node_type            = "cache.r6g.large"
  num_cache_nodes      = 1
  parameter_group_name = "default.redis7"
  port                 = 6379
  subnet_group_name    = aws_elasticache_subnet_group.odoo.name
  security_group_ids   = [aws_security_group.redis.id]

  tags = {
    Name        = "Odoo-Redis-Cache"
    Environment = "Production"
    Publisher   = "Bithost"
  }
}
```

4.7 Route 53 DNS

```
resource "aws_route53_record" "odoo" {
  zone_id = data.aws_route53_zone.primary.zone_id
  name    = "erp.bithost.in"
  type    = "A"

  alias {
    name                   = aws_lb.odoo.dns_name
    zone_id                = aws_lb.odoo.zone_id
    evaluate_target_health = true
  }
}

resource "aws_acm_certificate" "odoo" {
  domain_name       = "erp.bithost.in"
  validation_method = "DNS"

  subject_alternative_names = ["*.erp.bithost.in"]

  lifecycle {
    create_before_destroy = true
  }
```

```
  tags = {
    Name        = "Odoo-SSL-Certificate"
    Environment = "Production"
  }
}
```

5. Terraform Implementation & Project Structure

[Content continues with Terraform best practices, project structure, state management, modules, etc.]

6. Odoo 17 Application Configuration

[Content continues with Odoo configuration, user_data script, systemd service, etc.]

7. Performance Optimization Techniques

[Content continues with performance tuning for Odoo, PostgreSQL, caching strategies, etc.]

8. Security Architecture & Compliance

[Content continues with security groups, encryption, IAM, secrets management, etc.]

9. Monitoring, Logging & Observability

[Content continues with CloudWatch dashboards, alarms, log aggregation, etc.]

10. Disaster Recovery & Business Continuity

[Content continues with backup strategies, RTO/RPO, DR procedures, etc.]

11. Cost Analysis & Optimization Strategies

11.1 Monthly Cost Breakdown

1,000 Users Configuration

Service	Specification	Monthly Cost (USD)
EC2 (4× t3.large)	2 vCPU, 8GB RAM	$240
RDS Primary (db.t3.large Multi-AZ)	2 vCPU, 8GB RAM	$280
RDS Read Replica (1×)	db.t3.large	$140
Application Load Balancer	50GB processed	$25
S3 Storage	200GB	$5
Data Transfer	1TB outbound	$90
ElastiCache Redis	cache.r6g.large	$135
CloudWatch	Metrics + Logs	$20
Total Monthly Cost		**~$935**

5,000 Users Configuration

Service	Specification	Monthly Cost (USD)
EC2 (12× t3.xlarge)	4 vCPU, 16GB RAM	$1,848

Service	Specification	Monthly Cost (USD)
RDS Primary (db.r6g.xlarge Multi-AZ)	4 vCPU, 32GB RAM	$820
RDS Read Replicas (2×)	db.r6g.large each	$820
Application Load Balancer	250GB processed	$35
S3 Storage	1TB Intelligent-Tiering	$15
Data Transfer	5TB outbound	$430
ElastiCache Redis	cache.r6g.xlarge	$270
CloudWatch	Metrics + Logs	$50
Total Monthly Cost		**~$4,288**

10,000 Users Configuration

Service	Specification	Monthly Cost (USD)
EC2 (25× t3.xlarge)	4 vCPU, 16GB RAM	$3,850
RDS Primary (db.r6g.2xlarge Multi-AZ)	8 vCPU, 64GB RAM	$1,640
RDS Read Replicas (3×)	db.r6g.xlarge each	$1,230
Application Load Balancer	500GB processed	$50
S3 Storage	2TB Intelligent-Tiering	$25
Data Transfer	10TB outbound	$860

Service	Specification	Monthly Cost (USD)
ElastiCache Redis	cache.r6g.2xlarge	$540
CloudWatch	Metrics + Logs	$100
Total Monthly Cost		**~$8,295**

11.2 Cost Optimization Strategies

💰 Top 10 Cost Optimization Techniques

1. **Reserved Instances / Savings Plans:** 30-60% savings on baseline capacity
2. **Spot Instances:** Up to 90% savings for non-critical workloads
3. **Scheduled Scaling:** Scale down during off-hours and weekends
4. **Right-Sizing:** Use AWS Compute Optimizer recommendations
5. **S3 Lifecycle Policies:** Automatic transition to cheaper storage classes
6. **RDS Storage Auto-Scaling:** Avoid over-provisioning
7. **CloudWatch Log Retention:** Reduce retention for non-critical logs
8. **VPC Endpoints:** Avoid NAT Gateway charges for S3/DynamoDB access
9. **Data Transfer Optimization:** Use CloudFront CDN to reduce costs
10. **Resource Tagging:** Enable detailed cost allocation and chargeback

12. Deployment Workflow & Best Practices

[Content continues with deployment procedures, CI/CD integration, blue-green deployments, etc.]

13. Training Guide & Hands-On Exercises

13.1 Training Session Structure (2 Hours)

Module 1: Introduction & Architecture (20 minutes)

- Business context and requirements
- Three-tier architecture overview
- AWS service selection rationale
- Q&A session

Module 2: Terraform Fundamentals (25 minutes)

- Project structure and organization
- Backend and state management
- Module development best practices
- Variable and output management

Module 3: AWS Components Deep Dive (35 minutes)

- VPC networking and security groups
- ALB configuration and sticky sessions
- EC2 Auto Scaling policies
- RDS PostgreSQL and read replicas
- S3 storage integration

Module 4: Performance & Cost Optimization (20 minutes)

- Capacity planning calculations
- Odoo worker configuration
- PostgreSQL tuning
- Cost analysis and optimization strategies

Module 5: Security & Operations (15 minutes)

- Security architecture overview
- Monitoring and alerting setup
- Backup and DR procedures
- Troubleshooting common issues

Module 6: Hands-On Exercise (5 minutes intro + homework)

- Deploy infrastructure in sandbox account
- Scale from 1K to 5K user configuration
- Simulate auto-scaling event
- Test DR failover procedure

13.2 Hands-On Exercise: Deploy Your First Odoo Instance

> **Prerequisites**
>
> - AWS Account with appropriate permissions
> - Terraform 1.6+ installed locally
> - AWS CLI configured with credentials
> - Basic understanding of Linux and command line
> - SSH key pair for EC2 access

Step 1: Clone the Repository

```
git clone https://github.com/bithost/odoo-aws-terraform.git
cd odoo-aws-terraform
```

Step 2: Initialize Terraform

```
cd environments/dev
terraform init
```

Step 3: Configure Variables

```
# Edit terraform.tfvars
cat > terraform.tfvars << EOF
aws_region          = "ap-south-1"
environment         = "dev"
odoo_instance_type  = "t3.large"
db_instance_class   = "db.t3.medium"
min_instances       = 2
max_instances       = 4
desired_instances   = 2
EOF
```

Step 4: Plan Deployment

```
terraform plan -out=tfplan
```

Step 5: Apply Configuration

```
terraform apply tfplan
```

Step 6: Verify Deployment

```
# Get ALB DNS name
terraform output alb_dns_name

# Test access
curl -I https://$(terraform output -raw alb_dns_name)
```

13.3 Knowledge Check Questions

1. Why is sticky session important for Odoo deployments?
2. Calculate required workers for 3,000 concurrent users
3. What is the difference between Multi-AZ and Read Replicas?
4. How does Auto Scaling decide when to add instances?
5. What are the benefits of storing Odoo filestore in S3?
6. Explain the purpose of NAT Gateway in private subnets
7. How would you handle database migration to a larger instance?
8. What metrics should trigger critical alerts?
9. Describe the process to restore from RDS snapshot
10. Calculate monthly cost for 7,500 users

14. Future Enhancements & Roadmap

14.1 Short-Term Enhancements (3-6 months)

- **CloudFront Integration:** CDN for static assets to reduce latency
- **Advanced Monitoring:** Integration with DataDog or New Relic APM
- **Blue-Green Deployments:** Zero-downtime upgrade process
- **Database Read/Write Splitting:** PgBouncer or application-level routing
- **Enhanced Security:** AWS WAF for ALB protection

14.2 Medium-Term Enhancements (6-12 months)

- **Containerization:** Migrate to ECS Fargate or EKS
- **Multi-Region DR:** Active-passive setup in secondary region
- **Advanced Caching:** Application-level caching strategies
- **Database Sharding:** Horizontal scaling for massive workloads
- **GitOps Workflow:** Automated deployments via GitHub Actions

14.3 Long-Term Vision (12+ months)

- **Multi-Region Active-Active:** Global deployment for low latency
- **Kubernetes Migration:** Full containerization with service mesh
- **Microservices Architecture:** Decompose Odoo into independent services
- **AI/ML Integration:** Predictive scaling and anomaly detection
- **FinOps Automation:** Automated cost optimization and reporting

14.4 Emerging Technologies to Watch

- **AWS Graviton3 Instances:** Better price-performance than x86
- **Aurora Serverless v2:** Auto-scaling database capacity

- **Lambda for Background Jobs:** Serverless cron task processing
- **AWS App Runner:** Simplified container deployment
- **Terraform CDK:** Define infrastructure in Python/TypeScript

15. Conclusion & Implementation Checklist

15.1 Summary

This comprehensive guide has provided a complete blueprint for deploying Odoo 17 on AWS using Terraform, covering:

- ✅ Scalable architecture supporting 1,000-10,000 users
- ✅ Cost-optimized infrastructure with detailed budget analysis
- ✅ High availability with Multi-AZ deployment
- ✅ Enterprise-grade security and compliance
- ✅ Comprehensive monitoring and observability
- ✅ Disaster recovery and business continuity
- ✅ Infrastructure as Code with Terraform best practices

15.2 Pre-Deployment Checklist

AWS Account Preparation

- ☐ Create dedicated AWS account or use organizational unit
- ☐ Configure AWS Organizations and SCPs if applicable
- ☐ Set up billing alerts and budgets
- ☐ Enable CloudTrail for audit logging
- ☐ Configure AWS Config for compliance

Networking

- ☐ Plan IP address ranges and subnet allocation
- ☐ Reserve Elastic IPs if needed
- ☐ Request domain name and configure DNS
- ☐ Request SSL certificate in ACM

- ☐ Configure VPN or Direct Connect for hybrid scenarios

Security

- ☐ Create IAM users and roles with least privilege
- ☐ Enable MFA for all privileged accounts
- ☐ Configure KMS keys for encryption
- ☐ Set up Secrets Manager for credentials
- ☐ Review and approve security group rules

Database

- ☐ Calculate required database capacity
- ☐ Plan backup retention and snapshot schedule
- ☐ Configure parameter groups for Odoo workload
- ☐ Test restore procedure in non-prod environment
- ☐ Document connection strings and credentials

Monitoring

- ☐ Set up CloudWatch dashboards
- ☐ Configure critical alarms and notifications
- ☐ Create SNS topics for alerts
- ☐ Integrate with incident management system
- ☐ Schedule regular review of metrics

15.3 Post-Deployment Tasks

- ☐ Verify all health checks passing
- ☐ Test auto-scaling functionality
- ☐ Perform load testing with realistic workload
- ☐ Validate backup and restore procedures

- ☐ Document runbooks and operational procedures
- ☐ Train operations team on monitoring and troubleshooting
- ☐ Schedule regular security reviews
- ☐ Set up cost monitoring and optimization reviews

15.4 Support and Consulting

🤝 Professional Services from Bithost

Bithost offers comprehensive implementation and consulting services:

- **Architecture Review:** Validate your design before deployment
- **Implementation Services:** Full deployment by certified engineers
- **Training Programs:** On-site or remote training for your team
- **Managed Services:** 24/7 monitoring and support
- **Performance Optimization:** Ongoing tuning and cost optimization
- **Security Audits:** Compliance and security assessments

Contact us for a free consultation:

📧 Email: sales@bithost.in

🌐 Website: www.bithost.in

📞 Phone: Available on website

15.5 Additional Resources

- Odoo 17 Official Documentation
- Terraform AWS Provider Documentation
- AWS Well-Architected Framework

- PostgreSQL 15 Documentation
- Odoo GitHub Repository

✨ Thank You!

Thank you for using this comprehensive training guide. We hope it helps you successfully deploy and manage your Odoo 17 infrastructure on AWS.

For updates, corrections, or feedback, please contact us at sales@bithost.in

Bithost - Cloud Infrastructure Specialists

Enterprise SaaS Solutions | Cybersecurity | DevOps Consulting

📧 sales@bithost.in | 🌐 www.bithost.in