Cloud-Ops Central

Phase 2: Org Setup & Configuration

Goal - Establish a secure and properly configured Salesforce organization foundation that supports multi-cloud operations with appropriate user access controls and organizational settings.

Salesforce Editions

Developer Edition will be used for this project as it provides all necessary features including custom objects, Apex development, Lightning components, and API integration capabilities without licensing costs.

Company Profile Setup

Configure the organization profile with:

- **Company Name**: CloudOps Central Solutions
- **Business Type**: Technology/Cloud Services
- **Industry**: Information Technology & Cloud Computing
- **Locale Settings**: English (United States)
- **Time Zone**: (GMT-08:00) Pacific Standard Time

Business Hours & Holidays

- **Standard Business Hours**: Monday-Friday, 8:00 AM - 6:00 PM PST
- **Holiday Schedule**: Major US holidays (New Year's Day, Memorial Day, Independence Day, Labor Day, Thanksgiving, Christmas)
- **Support Hours**: 24/7 for critical cloud infrastructure issues
- **Maintenance Windows**: Sunday 2:00 AM - 4:00 AM PST for system updates

Fiscal Year Settings

- **Fiscal Year**: January 1 - December 31 (Calendar Year)
- **Quarter Structure**: Q1 (Jan-Mar), Q2 (Apr-Jun), Q3 (Jul-Sep), Q4 (Oct-Dec)
- This aligns with typical cloud billing cycles and budget planning processes

User Setup & Licenses

- **System Administrator**: Full platform access
- **Cloud Architect**: Platform license for strategic planning
- **DevOps Engineer**: Platform license for deployment management
- **Site Reliability Engineer**: Platform license for monitoring
- **Cloud Financial Analyst**: Salesforce license for cost analysis
- **Manager**: Salesforce license for dashboard viewing

Profiles

- **Cloud Admin Profile**: Full administrative access to all objects
- **Cloud Architect Profile**: Read/write infrastructure, limited admin
- **DevOps Engineer Profile**: Full deployment access, limited cost data
- **SRE Profile**: Monitoring and incident management
- **FinOps Analyst Profile**: Full cost access, read-only technical data
- **Cloud Manager Profile**: Dashboard access and approvals

Roles

- Chief Technology Officer (CTO)
- VP of Cloud Operations
- Cloud Architecture Team Lead
- DevOps Team Lead
- SRE Team Lead
- FinOps Manager
- Senior Cloud Engineer
- Junior Cloud Engineer

Permission Sets

- **Advanced API Access**: For external system integrations
- **Cost Management Plus**: Enhanced budget management
- **Emergency Response**: Elevated permissions for critical incidents

- **Audit Access**: Compliance and security reviews

OWD (Organization-Wide Defaults)

- **Cloud Provider**: Public Read Only
- **Cloud Resource**: Private
- **Cost Record**: Private
- **Deployment**: Public Read Only
- **Incident**: Private

Sharing Rules

- Share Cloud Resources with DevOps Team (Read/Write)
- Share Cost Records with FinOps Team (Read/Write)
- Share Critical Incidents with SRE Team (Read/Write)
- Share Budget Data with Management (Read Only)

Login Access Policies

- **Multi-Factor Authentication**: Required for all users
- **Session Timeout**: 2 hours (regular), 8 hours (admin)
- **IP Restrictions**: Corporate network and approved VPN
- **Login Hours**: 24/7 for SRE, business hours for others
- **Password Complexity**: 8+ characters, mixed case, numbers, special characters

Dev Org Setup

- **Primary Development Org**: Full feature development
- **Integration Sandbox**: API testing
- **UAT Sandbox**: Business user validation

Sandbox Usage

- **Developer Pro Sandbox**: Complex feature development
- **Partial Copy Sandbox**: User training
- **Full Copy Sandbox**: Complete system testing

- **Refresh Schedule**: Monthly (dev), Quarterly (testing)

Deployment Basics

- **Change Sets**: Configuration changes between environments
- **Version Control**: Git repository for code tracking
- **Release Management**: Structured deployment with approval gates
- **Rollback Procedures**: Documented process for reversions

Phase 3: Data Modeling & Relationships

Goal - Design and implement a comprehensive data model that accurately represents multi-cloud infrastructure, cost management, and DevOps operations while maintaining data integrity and supporting complex business processes.

Standard & Custom Objects

Custom Objects Created:

1. **Cloud Provider** (Custom)
   - Purpose: Master data for cloud service providers
   - Key Fields: Provider Name, API Endpoint, Authentication Type, Region List
   - Record Types: Public Cloud, Private Cloud, Hybrid Cloud
2. **Cloud Resource** (Custom)
   - Purpose: Individual cloud infrastructure components
   - Key Fields: Resource Name, Type, Provider, Region, Status, Cost Center
   - Record Types: Compute (EC2, VM), Storage (S3, Blob), Database (RDS, SQL), Network
3. **Cost Record** (Custom)
   - Purpose: Detailed cost and usage tracking
   - Key Fields: Billing Period, Usage Amount, Cost Amount, Currency
   - Record Types: Actual Cost, Budgeted Cost, Forecast Cost

4. **Deployment** (Custom)
    - Purpose: CI/CD pipeline and infrastructure deployment tracking
    - Key Fields: Deployment Name, Environment, Status, Start Time, End Time
    - Record Types: Application Deployment, Infrastructure Deployment
5. **Cloud Account** (Custom)
    - Purpose: Cloud provider account/subscription management
    - Key Fields: Account ID, Provider, Billing Contact, Environment Type
    - Record Types: Production Account, Development Account, Testing Account
6. **Budget** (Custom)
    - Purpose: Budget planning and threshold management
    - Key Fields: Budget Name, Amount, Period, Alert Thresholds
    - Record Types: Monthly Budget, Quarterly Budget, Project Budget

Fields

Cloud Resource Fields:

- Resource ID (External ID) - Unique cloud provider identifier
- CPU Utilization (Percent) - Processor usage
- Memory Utilization (Percent) - Memory usage
- Network I/O (Number) - Data transfer metrics
- Storage Used (Number) - Storage consumption
- Last Monitored (DateTime) - Most recent check
- Optimization Score (Number) - Efficiency rating (1-100)

Cost Record Fields:

- Unit Cost (Currency) - Cost per usage unit
- Reserved Instance Discount (Percent) - RI savings
- Volume Discount (Percent) - Volume pricing reduction
- Tax Amount (Currency) - Applied taxes

- Cost Category (Picklist) - Compute, Storage, Network, Database, Security

Record Types

Cloud Resource Record Types:

- **Compute Resources**: EC2 instances, VMs, container services
- **Storage Resources**: Object storage, block storage, databases
- **Network Resources**: Load balancers, CDN, VPN
- **Security Resources**: Firewalls, identity services, certificates

Deployment Record Types:

- **Blue/Green Deployment**: Zero-downtime strategy
- **Rolling Deployment**: Gradual rollout
- **Canary Deployment**: Limited testing rollout
- **Infrastructure as Code**: Terraform/CloudFormation deployments

Page Layouts

Cloud Resource Layout:

- **Header**: Resource Name, Provider, Status, Region
- **Details Section**: Technical specifications
- **Cost Section**: Current costs, budget allocation, optimization
- **Monitoring Section**: Performance metrics, alerts, health
- **Related Lists**: Deployments, cost records, incidents

Cost Record Layout:

- **Header**: Billing Period, Total Cost, Budget Status
- **Cost Breakdown**: Service-wise distribution
- **Usage Metrics**: Consumption patterns
- **Optimization**: Savings opportunities
- **Approval Section**: Budget approvals and allocation

Compact Layouts

- **Cloud Resource**: Name, Provider, Status, Monthly Cost
- **Deployment**: Name, Environment, Status, Last Deploy Date
- **Budget**: Name, Current Spend, Budget Limit, Utilization %

Schema Builder

The schema implements a hub-and-spoke model:

- **Hub**: Cloud Provider (center)
- **Connected Objects**: Cloud Accounts → Cloud Resources → Cost Records
- **Supporting Objects**: Budgets, Deployments link to resources

Lookup vs Master-Detail vs Hierarchical Relationships

Master-Detail Relationships:

- Cloud Provider → Cloud Account (Provider controls account)
- Cloud Account → Cloud Resource (Account deletion removes resources)
- Budget → Cost Record (Budget controls cost tracking)

Lookup Relationships:

- Cloud Resource → Deployment (flexible association)
- Cost Record → Cloud Resource (optional cost allocation)
- User → Cloud Resource (ownership assignment)

Hierarchical Relationships:

- Cloud Account → Parent Cloud Account (organizational structure)
- Budget → Parent Budget (budget hierarchy)

Junction Objects

Resource Deployment Junction:

- Purpose: Many-to-many relationship between Resources and Deployments
- Use Case: Single deployment affects multiple resources
- Fields: Deployment Impact, Resource Status Change, Rollback Capability

Cost Allocation Junction:

- Purpose: Distribute costs across departments/projects
- Use Case: Shared resources with split responsibility
- Fields: Allocation Percentage, Responsible Department, Billing Code

External Objects

Cloud Provider API Data:

- Purpose: Real-time data from cloud APIs without storage
- Use Case: Live resource status, current pricing, availability
- Connection: REST API callouts

Third-Party Monitoring Tools:

- Purpose: Integration with Datadog, New Relic
- Use Case: Performance metrics and alerting
- Connection: OData or REST API integration

Phase 4: Process Automation (Admin)

Goal - Implement intelligent automation workflows that streamline cloud operations, enforce governance policies, and provide proactive cost management while maintaining system reliability.

Validation Rules

Cloud Resource Validation Rules:

- **Resource Naming Convention**: REGEX validation for company naming standards (Environment-Service-Team-Version)
- **Budget Threshold Check**: Prevents resource creation if estimated cost exceeds remaining budget
- **Region Compliance**: Ensures resources created only in approved geographic regions
- **Environment Tagging**: Requires proper environment tags (Production, Staging, Development, Testing)

Cost Record Validation Rules:

- **Future Date Prevention**: Blocks cost records with dates >30 days in future
- **Negative Cost Check**: Prevents negative amounts except for refunds/credits
- **Currency Consistency**: Ensures cost currency matches account default currency
- **Billing Period Overlap**: Prevents overlapping periods for same resource

Deployment Validation Rules:

- **Production Deployment Hours**: Restricts deployments to approved maintenance windows
- **Approval Requirement**: Mandates approval for deployments affecting >50 resources or costing >$10,000
- **Rollback Plan Required**: Ensures rollback procedures documented for production
- **Environment Progression**: Enforces deployment path (Dev → Test → Staging → Production)

Workflow Rules

Resource Status Workflow:

- **Trigger**: Resource Status changes to "Critical" or "Failed"
- **Action**: Email alert to SRE team, create high-priority case
- **Time-based**: If status remains Critical for 1 hour, escalate to management

Budget Alert Workflow:

- **Trigger**: Budget utilization exceeds 80% threshold
- **Action**: Email notification to budget owner and finance team
- **Time-based**: Daily reminders when utilization exceeds 95%

Process Builder

Cost Optimization Process:

- **Trigger**: New/Updated Cost Record
- **Criteria 1**: Cost increase >20% from previous month
  - Create optimization review task for FinOps team
  - Update resource with "Review Required" flag

- **Criteria 2**: Resource utilization <30% for 7 consecutive days
    - Generate rightsizing recommendation
    - Send cost optimization opportunity email

Resource Lifecycle Process:

- **Trigger**: Cloud Resource created or updated
- **Criteria 1**: Resource Age >90 days AND Status = "Unused"
    - Create decommission review task
    - Send cleanup notification to owner
- **Criteria 2**: Resource created in Production
    - Create security review task
    - Apply production sharing rules

Approval Process

High-Cost Deployment Approval:

- **Entry Criteria**: Estimated cost >$5,000 OR affects >25 production resources
- **Approval Steps**:
    1. **Technical Review**: Senior Cloud Architect (auto-approve if <$10,000)
    2. **Financial Review**: FinOps Manager (required for all)
    3. **Executive Approval**: VP Cloud Operations (required if >$25,000)
- **Final Actions**: Update status, send notifications
- **Rejection Actions**: Create feedback task, notify submitter

Budget Override Approval:

- **Entry Criteria**: Requested budget increase >15% of current allocation
- **Approval Steps**:
    1. **Department Manager**: Immediate supervisor
    2. **Finance Review**: CFO or Finance Director
    3. **Executive Sign-off**: CTO (if increase >$50,000)

- **Approval Actions**: Update budget limits, recalculate thresholds

Flow Builder

Screen Flows:

Resource Provisioning Flow:

- **Screen 1**: Resource Type Selection (Compute, Storage, Database, Network)
- **Screen 2**: Configuration Details (Size, Region, Environment, Tags)
- **Screen 3**: Cost Estimation Display with budget impact
- **Screen 4**: Approval Requirements (if thresholds exceeded)
- **Actions**: Create resource, generate estimates, initiate approvals

Cost Analysis Flow:

- **Screen 1**: Analysis Period (Monthly, Quarterly, Custom)
- **Screen 2**: Filter Options (Provider, Department, Environment)
- **Screen 3**: Results Display with charts and recommendations
- **Actions**: Generate reports, export data, create optimization tasks

Record-Triggered Flows:

Incident Response Flow:

- **Trigger**: Cloud Resource Status → "Critical" or "Down"
- **Actions**:
  - Create incident record with severity
  - Send SMS/email alerts to on-call SRE team
  - Create Slack channel for coordination
  - Start incident timer for SLA tracking
  - Generate impact assessment

Budget Threshold Flow:

- **Trigger**: Cost Record update causes budget threshold breach
- **Decision Logic**:

    - 75%: Warning email to budget owner
    - 90%: Create review task, notify finance
    - 100%: Trigger approval for budget increase, halt non-critical resources
    - 110%: Executive escalation, emergency review

Scheduled Flows:

Daily Resource Health Check:

- **Schedule**: Daily at 6:00 AM PST
- **Actions**:
    - Query all active resources for metrics
    - Identify underutilized or overutilized resources
    - Generate optimization recommendations
    - Create daily health report
    - Update resource health scores

Monthly Cost Reconciliation:

- **Schedule**: 1st of each month at 2:00 AM PST
- **Actions**:
    - Import latest billing data from providers
    - Reconcile actual vs. estimated costs
    - Calculate budget variances and forecasts
    - Generate chargeback reports
    - Create monthly executive summary

Auto-launched Flows:

Resource Tagging Enforcement:

- **Trigger**: Called from Apex after resource creation
- **Actions**:
    - Validate required tags (Environment, CostCenter, Owner, Project)

    - Apply default tags based on resource type
    - Create tag compliance score
    - Generate remediation tasks for non-compliant resources

Deployment Impact Analysis:

- **Trigger**: Called from deployment approval process
- **Actions**:
    - Analyze affected resources and dependencies
    - Calculate business impact score
    - Estimate deployment duration and risk
    - Generate deployment checklist
    - Return assessment to approval process

Email Alerts

Cost Threshold Alerts:

- **Template**: "Budget Alert - {Budget Name} at {Utilization}%"
- **Recipients**: Budget owner, finance team, department manager
- **Content**: Current spend, remaining budget, trend analysis
- **Frequency**: Immediate for critical, daily digest for warnings

Resource Performance Alerts:

- **Template**: "Resource Performance Issue - {Resource Name}"
- **Recipients**: Resource owner, SRE team, DevOps engineer
- **Content**: Performance metrics, historical trends, recommended actions
- **Escalation**: Management notification if issue persists >2 hours

Field Updates

Automated Status Updates:

- **Resource Health Score**: Calculated based on performance metrics and cost efficiency
- **Budget Utilization Percentage**: Real-time calculation of spend vs. allocation

- **Last Optimization Date**: Updated when recommendations applied
- **Compliance Score**: Based on tagging, security, and governance policies

Tasks

Optimization Review Tasks:

- **Subject**: "Review Cost Optimization Opportunities - {Resource Name}"
- **Assigned To**: FinOps team member by resource type
- **Priority**: Normal (High if potential savings >$1,000/month)
- **Due Date**: 5 business days

Security Review Tasks:

- **Subject**: "Security Configuration Review - {Resource Name}"
- **Assigned To**: Security team member
- **Priority**: High for production, Normal for development
- **Due Date**: 2 business days (production), 5 days (non-production)

Custom Notifications

Slack Integration Notifications:

- **Critical Incidents**: Immediate to #sre-alerts channel
- **Deployment Status**: Updates to #devops-deployments channel
- **Cost Alerts**: Notifications to #finops-alerts channel
- **Optimization Opportunities**: Weekly digest to #cost-optimization

Mobile Push Notifications:

- **Critical System Alerts**: Immediate to on-call team mobile devices
- **Approval Requests**: Push for pending executive approvals
- **Budget Overruns**: Push to finance team for threshold breaches

Phase 5: Apex Programming (Developer)

Goal - Develop robust, scalable Apex solutions that handle complex cloud operations logic, integrate with external cloud APIs, and provide advanced automation capabilities.

Classes & Objects

Core Apex Classes:

1. **CloudResourceManager**: CRUD operations, lifecycle management, performance calculations
2. **CloudProviderAPIService**: Generic cloud API framework, authentication, rate limiting
3. **CostCalculationEngine**: Complex cost calculations, budget allocation, ROI analysis
4. **DeploymentOrchestrator**: CI/CD integration, deployment validation, rollback procedures

Apex Triggers (before/after insert/update/delete)

CloudResourceTrigger:

- **Before Insert**: Validate naming, apply default tags, calculate cost estimates
- **After Insert**: Create monitoring setup, generate security reviews, update account totals
- **Before Update**: Validate status changes, check approval requirements
- **After Update**: Process status notifications, update cost projections, trigger optimization
- **Before Delete**: Validate decommission approval, backup configuration
- **After Delete**: Clean up related records, update capacity planning

CostRecordTrigger:

- **Before Insert**: Validate billing periods, apply currency conversions
- **After Insert**: Update budget utilization, trigger threshold alerts
- **Before Update**: Prevent historical data modification
- **After Update**: Recalculate budget status, update optimization opportunities

DeploymentTrigger:

- **Before Insert**: Validate deployment windows, check dependencies
- **After Insert**: Initiate deployment pipeline, create monitoring entries

- **Before Update**: Validate status transitions, check rollback requirements
- **After Update**: Process completion workflows, update resource configurations

Trigger Design Pattern

Handler Pattern Implementation:

```
// Trigger
trigger CloudResourceTrigger on Cloud_Resource__c (before insert, after insert, before update, after update) {
    CloudResourceTriggerHandler.execute();
}

// Handler Class
public class CloudResourceTriggerHandler {
    public static void execute() {
        if (Trigger.isBefore && Trigger.isInsert) beforeInsert();
        if (Trigger.isAfter && Trigger.isInsert) afterInsert();
        // Additional trigger contexts...
    }

    private static void beforeInsert() {
        for (Cloud_Resource__c resource : (List<Cloud_Resource__c>) Trigger.new) {
            // Validation and default value logic
        }
    }
}
```

SOQL & SOSL

Complex SOQL Queries:

```
// Resource utilization analysis with cost correlation
List<Cloud_Resource__c> underUtilizedResources = [
    SELECT Id, Name, CPU_Utilization__c, Memory_Utilization__c,
           (SELECT Amount__c FROM Cost_Records__r WHERE Billing_Period__c = THIS_MONTH)
    FROM Cloud_Resource__c
    WHERE CPU_Utilization__c < 30
    AND Status__c = 'Active'
    ORDER BY Cost_Records__r.Amount__c DESC
];
```

SOSL Global Search:

```
// Search across cloud infrastructure
List<List<SObject>> searchResults = [
    FIND 'production database' IN ALL FIELDS
    RETURNING Cloud_Resource__c(Name, Type__c), Deployment__c(Name, Environment__c)
];
```

Collections: List, Set, Map

Advanced Collection Usage:

```
Map<String, List<Cloud_Resource__c>> resourcesByType = new Map<String, List<Cloud_Resource__c>>();
Set<Id> optimizationCandidates = new Set<Id>();
Map<Id, Decimal> costSavingsOpportunity = new Map<Id, Decimal>();


for (Cloud_Resource__c resource : resources) {
    if (!resourcesByType.containsKey(resource.Type__c)) {
        resourcesByType.put(resource.Type__c, new List<Cloud_Resource__c>());
    }
    resourcesByType.get(resource.Type__c).add(resource);
}
```

Control Statements

Business Logic Implementation:

```
public Decimal calculateOptimizedCost(Cloud_Resource__c resource) {
    Decimal optimizedCost = resource.Current_Cost__c;

    if (resource.CPU_Utilization__c < 25) {
        optimizedCost *= 0.5; // Downsize recommendation
    } else if (resource.CPU_Utilization__c > 80) {
        optimizedCost *= 1.5; // Upsize recommendation
    }

    return optimizedCost;
}
```

Batch Apex

CloudResourceOptimizationBatch:

```
public class CloudResourceOptimizationBatch implements Database.Batchable<sObject> {

    public Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator([
            SELECT Id, Name, CPU_Utilization__c, Current_Cost__c
            FROM Cloud_Resource__c
            WHERE Last_Optimization_Date__c < LAST_N_DAYS:30
        ]);
    }
    public void execute(Database.BatchableContext bc, List<Cloud_Resource__c> resources) {
        List<Optimization_Opportunity__c> opportunities = new
List<Optimization_Opportunity__c>();

        for (Cloud_Resource__c resource : resources) {
            // Analyze and create optimization recommendations
            if (resource.CPU_Utilization__c < 30) {
                Optimization_Opportunity__c opp = new Optimization_Opportunity__c(
                    Cloud_Resource__c = resource.Id,
                    Type__c = 'Rightsizing',
                    Potential_Savings__c = resource.Current_Cost__c * 0.5
                );
                opportunities.add(opp);
            }
        }

        if (!opportunities.isEmpty()) insert opportunities;
    }

    public void finish(Database.BatchableContext bc) {
        // Send summary report
    }
}
```

Queueable Apex

CloudAPIIntegrationQueue:

```
public class CloudAPIIntegrationQueue implements Queueable, Database.AllowsCallouts {
    private List<Id> resourceIds;

    public CloudAPIIntegrationQueue(List<Id> resourceIds) {
        this.resourceIds = resourceIds;
    }

    public void execute(QueueableContext context) {
        // Process API calls for resource updates
        for (Id resourceId : resourceIds) {
            updateResourceMetrics(resourceId);
        }

        // Chain next batch if needed
        if (resourceIds.size() > 100) {
            List<Id> nextBatch = new List<Id>();
            for (Integer i = 100; i < Math.min(200, resourceIds.size()); i++) {
                nextBatch.add(resourceIds[i]);
            }
            System.enqueueJob(new CloudAPIIntegrationQueue(nextBatch));
        }
    }
}
```

Scheduled Apex

DailyCloudResourceHealthCheck:

```
public class DailyCloudResourceHealthCheck implements Schedulable {
   public void execute(SchedulableContext sc) {
       // Run daily health assessment
       Database.executeBatch(new CloudResourceOptimizationBatch(), 200);

       // Generate daily reports
       System.enqueueJob(new DailyCostReportGenerator());
   }
}

// Schedule: Daily at 6 AM PST
// System.schedule('Daily Health Check', '0 0 6 * * ?', new DailyCloudResourceHealthCheck());
```

Future Methods

API Integration Methods:

```
@future(callout=true)
public static void syncResourceDataWithProvider(Set<Id> resourceIds) {
   for (Id resourceId : resourceIds) {
       Cloud_Resource__c resource = [SELECT External_ID__c FROM Cloud_Resource__c WHERE
Id = :resourceId];

       // Make API callout
       HttpRequest req = new HttpRequest();
       req.setEndpoint('callout:AWS_Integration/instances/' + resource.External_ID__c);
       req.setMethod('GET');

       Http http = new Http();
       HttpResponse res = http.send(req);
```

```
        if (res.getStatusCode() == 200) {
            // Parse and update resource
            updateResourceFromAPI(resource, res.getBody());
        }
    }
}
```

Exception Handling

Comprehensive Error Management:

```
public class CloudResourceService {
    public static void createResource(Cloud_Resource__c resource) {
        try {
            // Validate and create resource
            validateResourceConfiguration(resource);
            insert resource;

        } catch (CloudProviderException cpe) {
            System.debug('Cloud Provider Error: ' + cpe.getMessage());
            createErrorLog('CLOUD_API_ERROR', cpe.getMessage(), resource.Id);
            throw new AuraHandledException('Unable to create resource');

        } catch (DmlException dme) {
            System.debug('Database Error: ' + dme.getMessage());
            createErrorLog('DATABASE_ERROR', dme.getMessage(), resource.Id);
            throw new AuraHandledException('Database error occurred');
        } catch (Exception e) {
            System.debug('Unexpected Error: ' + e.getMessage());
            createErrorLog('SYSTEM_ERROR', e.getMessage(), resource.Id);
            throw new AuraHandledException('Unexpected error. Contact admin.');
        }
    }
}
```

Test Classes

CloudResourceManagerTest:

```
@isTest
public class CloudResourceManagerTest {

    @TestSetup
    static void setupTestData() {
        Cloud_Provider__c provider = new Cloud_Provider__c(Name = 'Test AWS');
        insert provider;

        Cloud_Account__c account = new Cloud_Account__c(
            Name = 'Test Account',
            Cloud_Provider__c = provider.Id
        );
        insert account;
    }

    @isTest
    static void testResourceCreation() {
        Cloud_Account__c account = [SELECT Id FROM Cloud_Account__c LIMIT 1];

        Cloud_Resource__c resource = new Cloud_Resource__c(
            Name = 'Test EC2',
            Type__c = 'Compute',
            Cloud_Account__c = account.Id,
            Status__c = 'Active'
        );

        Test.startTest();
        insert resource;
        Test.stopTest();

        Cloud_Resource__c created = [SELECT Health_Score__c FROM Cloud_Resource__c WHERE
Id = :resource.Id];
        System.assertNotEquals(null, created.Health_Score__c);
    }

    @isTest
    static void testBudgetThresholdAlert() {
        // Test budget alert workflow
        Cloud_Account__c account = [SELECT Id FROM Cloud_Account__c LIMIT 1];

        Budget__c budget = new Budget__c(Name = 'Test Budget', Amount__c = 10000);
        insert budget;
        Cost_Record__c costRecord = new Cost_Record__c(
            Budget__c = budget.Id,
            Amount__c = 9500, // 95% of budget
            Billing_Period__c = Date.today()
        );

        Test.startTest();
        insert costRecord;
        Test.stopTest();

        List<Task> alerts = [SELECT Id FROM Task WHERE Subject LIKE '%Budget Alert%'];
        System.assert(alerts.size() > 0, 'Budget alert should be created');
    }
}
```

Asynchronous Processing

Advanced Async Patterns:

- **Queueable Chaining**: Process large datasets in manageable chunks
- **Platform Events**: Real-time event publishing for resource alerts
- **Future + Queueable Combination**: Initial API calls with complex processing
- **Batch + Queueable**: Daily optimization batch triggers real-time notifications

Phase 6: User Interface Development

Goal - Create an intuitive, responsive, and feature-rich user interface that provides cloud engineers, managers, and executives with comprehensive visibility into multi-cloud operations.

Lightning App Builder

CloudOps Central Lightning App:

- **App Name**: CloudOps Central
- **App Type**: Standard Navigation (Tab)
- **Branding**: Cloud-themed color scheme (Blue: #1589FF, Green: #04844B)
- **Navigation Style**: Standard with utility bar
- **Mobile Support**: Fully optimized for Salesforce Mobile App

App Navigation Items:

- Home (Executive dashboard)
- Cloud Resources (Main management interface)
- Cost Analysis (Financial analytics)
- Deployments (CI/CD pipeline monitoring)
- Reports (Comprehensive reporting)
- Budgets (Budget planning and management)
- Optimization (Cost and performance recommendations)

Record Pages

Cloud Resource Record Page:

- **Header Section**: Resource name, status indicator, provider logo, last updated
- **Key Metrics Component**: CPU, Memory, Storage utilization with color-coded gauges
- **Cost Summary Component**: Monthly cost, budget allocation, optimization opportunities
- **Technical Details Tab**: Configuration specs, network settings, tags
- **Performance Tab**: Historical charts (24h, 7d, 30d), alerts, comparative analysis
- **Cost Tab**: Cost breakdown, historical trends, optimization recommendations

- **Related Lists**: Cost Records, Deployments, Incidents, Optimization opportunities

Budget Record Page:

- **Header Section**: Budget name, period, utilization percentage with progress bar
- **Budget Overview**: Allocated vs. spent with visual indicators
- **Department Allocation**: Cost center breakdown with drill-down
- **Forecast Component**: Predictive spending analysis with trend lines
- **Alert Settings Tab**: Threshold configuration and notification preferences
- **Related Lists**: Cost Records, Cloud Resources, Approval Requests

Deployment Record Page:

- **Header Section**: Deployment name, status, environment, deployment type
- **Pipeline Status**: Visual pipeline stages with status indicators
- **Impact Analysis**: Affected resources, estimated downtime, rollback plan
- **Timeline Tab**: Deployment progress with timestamps
- **Configuration Tab**: Infrastructure changes, code versions, approval chain
- **Related Lists**: Affected Resources, Approval Records, Validation Tasks

Tabs

Custom Tab Configuration:

- **Cloud Providers Tab**: Master data for AWS, Azure, GCP configurations
- **Cloud Accounts Tab**: Account and subscription management
- **Optimization Opportunities Tab**: Centralized cost and performance recommendations
- **Error Logs Tab**: Integration errors and troubleshooting (admin only)
- **API Usage Tab**: Cloud provider API consumption tracking

Home Page Layouts

Executive Home Page:

- **Top KPIs**: Total monthly spend, budget utilization, active resources, optimization savings
- **Cost Trend Chart**: 12-month spending trend with forecast

- **Resource Distribution Chart**: Resources by provider and type (pie chart)
- **Alert Summary**: Critical alerts requiring attention
- **Top Spending Resources**: Highest-cost resources with optimization opportunities
- **Recent Deployments**: Latest activities with success/failure indicators

Cloud Engineer Home Page:

- **Resource Health Dashboard**: Real-time status of monitored resources
- **Performance Alerts**: Resources requiring immediate attention
- **Deployment Queue**: Pending and active deployments with approval status
- **Cost Optimization Tasks**: Assigned optimization recommendations
- **Recent Activity Feed**: Timeline of resource changes and events

Manager Home Page:

- **Team Performance Metrics**: Deployment success rates, incident response times
- **Budget Status by Department**: Utilization across cost centers
- **Resource Utilization Trends**: Efficiency metrics and capacity planning
- **Approval Queue**: Pending budget and deployment approvals
- **Monthly Summary Reports**: Links to detailed analytical reports

Utility Bar

Quick Actions in Utility Bar:

- **Create Resource**: Quick provisioning with cost estimation
- **Emergency Alert**: Incident reporting with automatic escalation
- **Cost Calculator**: On-demand cost estimation for planning
- **API Status**: Real-time cloud provider API health monitoring
- **Quick Search**: Global search across resources, deployments, budgets

Utility Bar Components:

- **Live Chat**: Integration with Slack/Teams
- **Notification Center**: Real-time alerts and system notifications

- **Quick Notes**: Scratchpad for temporary notes
- **Recent Items**: Recently viewed resources and records

LWC (Lightning Web Components)

CloudResourceHealthGauge Component:

```
// Displays real-time resource health metrics
import { LightningElement, api, wire } from 'lwc';
import { getRecord } from 'lightning/uiRecordApi';

export default class CloudResourceHealthGauge extends LightningElement {
  @api recordId;

  @wire(getRecord, { recordId: '$recordId', fields: [CPU_FIELD, MEMORY_FIELD] })
  resource;

  get cpuUtilization() {
    return this.resource.data?.fields.CPU_Utilization__c.value || 0;
  }

  get healthStatus() {
    const score = this.healthScore;
    if (score >= 80) return 'Excellent';
    if (score >= 60) return 'Good';
    if (score >= 40) return 'Warning';
    return 'Critical';
  }
}
```

CostOptimizationRecommendations Component:

```
// Shows cost optimization opportunities with apply actions
import { LightningElement, api, wire } from 'lwc';
import getOptimizations from '@salesforce/apex/OptimizationController.getOptimizations';

export default class CostOptimizationRecommendations extends LightningElement {
    @api recordId;

    @wire(getOptimizations, { resourceId: '$recordId' })
    optimizations;

    handleApply(event) {
        const oppId = event.target.dataset.id;
        applyOptimization({ opportunityId: oppId })
            .then(() => this.showSuccessToast())
            .catch(error => this.showErrorToast(error));
    }
}
```

Multi-CloudProviderDashboard Component:

```
// Displays aggregated data across AWS, Azure, GCP
import { LightningElement, wire } from 'lwc';
import getMultiCloudSummary from
'@salesforce/apex/DashboardController.getMultiCloudSummary';

export default class MultiCloudProviderDashboard extends LightningElement {
    selectedProvider = 'All';
```

```
   @wire(getMultiCloudSummary, { provider: '$selectedProvider' })
   summaryData;

   get chartData() {
      return {
         labels: this.summaryData.data?.map(item => item.provider),
         datasets: [{ data: this.summaryData.data?.map(item => item.cost) }]
      };
   }
}
```

Apex with LWC

OptimizationController Class:

```
public with sharing class OptimizationController {

   @AuraEnabled(cacheable=true)
   public static List<Optimization_Opportunity__c> getOptimizations(Id resourceId) {
      return [
         SELECT Name, Recommendation_Type__c, Potential_Savings__c
         FROM Optimization_Opportunity__c
         WHERE Cloud_Resource__c = :resourceId
         AND Status__c = 'Open'
      ];
   }

   @AuraEnabled
```

```
    public static String applyOptimization(Id opportunityId) {
        Optimization_Opportunity__c opp = [SELECT Cloud_Resource__c FROM
Optimization_Opportunity__c WHERE Id = :opportunityId];

        Task task = new Task(
            Subject = 'Implement Optimization',
            WhatId = opp.Cloud_Resource__c,
            Status = 'Not Started'
        );
        insert task;

        return 'Success';
    }
}
```

Events in LWC

Resource Status Change Event Handling:

```
// Real-time monitoring using Platform Events
import { LightningElement } from 'lwc';
import { subscribe, onError } from 'lightning/empApi';

export default class ResourceStatusMonitor extends LightningElement {
    subscription = {};
    channelName = '/event/Resource_Alert__e';

    connectedCallback() {
        subscribe(this.channelName, -1, (response) => {
            this.handleResourceAlert(response.data.payload);
        });
    }

    handleResourceAlert(alertData) {
        const alertType = alertData.Alert_Type__c;
        const variant = alertType === 'Critical' ? 'error' : 'warning';
        this.dispatchEvent(new ShowToastEvent({
            title: `Resource Alert: ${alertType}`,
            message: alertData.Message__c,
            variant: variant
        }));
    }
}
```

Wire Adapters

Real-time Data Wire Adapter:

```
import { LightningElement, api, wire } from 'lwc';
import { refreshApex } from '@salesforce/apex';
import getRealTimeMetrics from '@salesforce/apex/ResourceController.getRealTimeMetrics';

export default class RealTimeResourceData extends LightningElement {
    @api recordId;
    wiredMetrics;

    @wire(getRealTimeMetrics, { resourceId: '$recordId' })
    metrics(result) {
        this.wiredMetrics = result;
    }

    connectedCallback() {
        // Refresh every 30 seconds
        setInterval(() => refreshApex(this.wiredMetrics), 30000);
    }
}
```

Imperative Apex Calls

Dynamic Cost Calculator:

```
import { LightningElement } from 'lwc';
import calculateCost from '@salesforce/apex/CostCalculator.calculateEstimatedCost';

export default class DynamicCostCalculator extends LightningElement {
    estimatedCost = 0;

    handleCalculate() {
        calculateCost({ configData: JSON.stringify(this.config) })
            .then(result => { this.estimatedCost = result; })
            .catch(error => { console.error(error); });
    }
}
```

Navigation Service

Navigation Between Records:

```
import { LightningElement } from 'lwc';
import { NavigationMixin } from 'lightning/navigation';
```

```
export default class ResourceNavigationManager extends NavigationMixin(LightningElement) {

    navigateToRelatedCosts() {
        this[NavigationMixin.Navigate]({
            type: 'standard__recordRelationshipPage',
            attributes: {
                recordId: this.recordId,
                relationshipApiName: 'Cost_Records__r',
                actionName: 'view'
            }
        });
    }

    navigateToOptimizationDashboard() {
        this[NavigationMixin.Navigate]({
            type: 'standard__navItemPage',
            attributes: { apiName: 'Optimization_Dashboard' }
        });
    }
}
```

Phase 7: Integration & External Access

Goal - Establish secure, reliable, and scalable integrations with cloud provider APIs, monitoring tools, and external systems to enable real-time data synchronization and automated operations.

Named Credentials

AWS API Integration:

- **Credential Name**: AWS_CloudOps_Integration
- **URL**: https://ec2.amazonaws.com
- **Authentication Protocol**: AWS Signature Version 4
- **AWS Access Key**: Securely stored with rotation policy
- **AWS Secret Key**: Encrypted storage
- **AWS Region**: us-west-2 (configurable)

Azure API Integration:

- **Credential Name**: Azure_CloudOps_Integration
- **URL**: https://management.azure.com
- **Authentication Protocol**: OAuth 2.0
- **Client ID**: Azure AD application registration
- **Client Secret**: Secured with automatic rotation
- **Tenant ID**: Organization-specific

Google Cloud API Integration:

- **Credential Name**: GCP_CloudOps_Integration
- **URL**: https://compute.googleapis.com
- **Authentication Protocol**: OAuth 2.0 with Service Account
- **Service Account**: cloudops-integration@project.iam.gserviceaccount.com
- **Private Key**: JSON key file stored securely

Monitoring Tools Integration:

- **Datadog**: API Key + Application Key authentication

- **New Relic**: API Key with Bearer token
- **ServiceNow**: OAuth 2.0 for ITSM integration

External Services

AWS EC2 Service Registration:

- **Service Name**: AWS EC2 Resource Management
- **Operations Exposed**:
    - DescribeInstances: Retrieve instance details
    - StartInstances: Power on instances
    - StopInstances: Power off instances
    - GetInstanceMetrics: CloudWatch performance data
- **Error Handling**: Automatic retry with exponential backoff
- **Timeout**: 120 seconds per operation

Azure Resource Manager Service:

- **Service Name**: Azure Resource Management
- **Operations Exposed**:
    - ListVirtualMachines: Enumerate VMs
    - GetVirtualMachine: Retrieve VM configuration
    - GetMetrics: Azure Monitor data
    - ListCosts: Cost management data
- **API Version**: 2023-09-01

Google Cloud Compute Service:

- **Service Name**: GCP Compute Engine Management
- **Operations Exposed**:
    - instances.list: List compute instances
    - instances.get: Get instance details
    - disks.list: Enumerate storage volumes

- **Pagination Support**: Automatic page token handling

Web Services (REST/SOAP)

CloudOps Central Public API (REST):

```
@RestResource(urlMapping='/api/v1/cloudresources/*')
global with sharing class CloudResourceAPI {

  @HttpGet
  global static CloudResourceResponse getResource() {
    String resourceId = RestContext.request.requestURI.substring(
      RestContext.request.requestURI.lastIndexOf('/') + 1
    );

    Cloud_Resource__c resource = [
      SELECT Id, Name, Status__c, CPU_Utilization__c
      FROM Cloud_Resource__c
      WHERE Id = :resourceId OR External_ID__c = :resourceId
    ];

    CloudResourceResponse response = new CloudResourceResponse();
    response.success = true;
    response.resource = resource;
    return response;
  }

  @HttpPost
  global static CloudResourceResponse createResource(CloudResourceRequest data) {
        Cloud_Resource__c newResource = new Cloud_Resource__c(
            Name = data.name,
            Type__c = data.resourceType,
            Provider__c = data.provider
        );
        insert newResource;

        CloudResourceResponse response = new CloudResourceResponse();
        response.success = true;
        response.resource = newResource;
        return response;
    }
}
```

SOAP Web Service (Legacy Integration):

```
global class CloudOpsCentralSOAPService {

    webservice static String getResourceStatus(String externalId) {
        Cloud_Resource__c resource = [
            SELECT Status__c
            FROM Cloud_Resource__c
            WHERE External_ID__c = :externalId
        ];
        return resource.Status__c;
    }
    webservice static String updateResourceMetrics(String externalId, Decimal cpuUtil, Decimal memUtil) {
        Cloud_Resource__c resource = [SELECT Id FROM Cloud_Resource__c WHERE External_ID__c = :externalId];
        resource.CPU_Utilization__c = cpuUtil;
        resource.Memory_Utilization__c = memUtil;
        update resource;
        return 'SUCCESS';
    }
}
```

Callouts

AWS EC2 API Callout:

```
public class AWSCloudCalloutService {

    public static HttpResponse describeInstances() {
        HttpRequest req = new HttpRequest();
        req.setEndpoint('callout:AWS_CloudOps_Integration/');
        req.setMethod('POST');
        req.setHeader('Content-Type', 'application/x-www-form-urlencoded');
        req.setBody('Action=DescribeInstances&Version=2016-11-15');

        Http http = new Http();
        return http.send(req);
    }

    public static void syncEC2Instances() {
        HttpResponse res = describeInstances();
        if (res.getStatusCode() == 200) {
            parseAndSyncEC2Data(res.getBody());
        }
    }

    private static void parseAndSyncEC2Data(String xmlResponse) {
        // Parse XML and upsert Cloud_Resource__c records
        Dom.Document doc = new Dom.Document();
        doc.load(xmlResponse);
        // Extract instance data and create records
    }
}
```

Azure Resource Manager Callout:

```
public class AzureCloudCalloutService {

    public static HttpResponse listVirtualMachines(String subscriptionId) {
        HttpRequest req = new HttpRequest();
        req.setEndpoint('callout:Azure_CloudOps_Integration/subscriptions/' +
                       subscriptionId + '/providers/Microsoft.Compute/virtualMachines?api-
version=2023-09-01');
        req.setMethod('GET');
        req.setHeader('Content-Type', 'application/json');

        Http http = new Http();
        return http.send(req);
    }

    private static void parseAzureVMResponse(String jsonResponse) {
        Map<String, Object> data = (Map<String, Object>) JSON.deserializeUntyped(jsonResponse);
        List<Object> vms = (List<Object>) data.get('value');

        // Create/update Cloud_Resource__c records
        List<Cloud_Resource__c> resources = new List<Cloud_Resource__c>();
        for (Object vmObj : vms) {
            Map<String, Object> vm = (Map<String, Object>) vmObj;
            Cloud_Resource__c resource = new Cloud_Resource__c(
                External_ID__c = (String) vm.get('id'),
                Name = (String) vm.get('name'),
                Provider__c = 'Azure'
            );
            resources.add(resource);
        }
        upsert resources External_ID__c;
    }
}
```

GCP Compute Engine Callout:

```
public class GCPCloudCalloutService {

    public static HttpResponse listInstances(String projectId, String zone) {
        HttpRequest req = new HttpRequest();
        req.setEndpoint('callout:GCP_CloudOps_Integration/compute/v1/projects/' +
                    projectId + '/zones/' + zone + '/instances');
        req.setMethod('GET');
        req.setHeader('Content-Type', 'application/json');

        Http http = new Http();
        return http.send(req);
    }
}
```

Platform Events

Resource_Alert__e Platform Event:

```
// Publishing Platform Events
public class ResourceAlertPublisher {

    public static void publishCriticalAlert(Id resourceId, String message) {
        Resource_Alert__e alert = new Resource_Alert__e(
            Resource_Id__c = resourceId,
            Alert_Type__c = 'Critical',
            Message__c = message,
            Severity__c = 'High',
            Timestamp__c = DateTime.now()
        );

        EventBus.publish(alert);
    }
}
```

```
// Subscribing to Platform Events
trigger ResourceAlertTrigger on Resource_Alert__e (after insert) {
    List<Case> cases = new List<Case>();

    for (Resource_Alert__e alert : Trigger.new) {
        if (alert.Severity__c == 'High') {
            Case c = new Case(
                Subject = 'Critical Alert: ' + alert.Alert_Type__c,
                Description = alert.Message__c,
                Priority = 'High'
            );
            cases.add(c);
        }
    }

    if (!cases.isEmpty()) insert cases;
}
```

Change Data Capture

Cloud_Resource__c Change Data Capture:

```
trigger CloudResourceChangeTrigger on Cloud_Resource__ChangeEvent (after insert) {

    for (Cloud_Resource__ChangeEvent event : Trigger.new) {
        List<String> changedFields = event.getChangedFields();

        // Handle status changes
        if (changedFields.contains('Status__c')) {
            String newStatus = (String) event.get('Status__c');
            if (newStatus == 'Critical') {
                // Trigger immediate response
                System.enqueueJob(new EmergencyResponseQueue(event.get('Id')));
            }
        }

        // Sync to external monitoring tools
        syncToExternalSystems(event.get('Id'));
    }
}
```

Salesforce Connect

External Object Configuration:

- **External Object**: AWS_EC2_Instance__x
- **External Data Source**: AWS CloudWatch API
- **Connection Type**: OData 4.0
- **Fields**: Instance_ID__c, Instance_Type__c, State__c, CPU_Utilization__c

External Data Query:

```
// Query external objects for live data
List<AWS_EC2_Instance__x> liveInstances = [
    SELECT Instance_ID__c, CPU_Utilization__c
    FROM AWS_EC2_Instance__x
    WHERE State__c = 'running' AND CPU_Utilization__c > 80
];

// Join with internal Salesforce data
```

```
List<Cloud_Resource__c> resources = [
    SELECT Id, (SELECT CPU_Utilization__c FROM AWS_EC2_Instance__xr)
    FROM Cloud_Resource__c
    WHERE Provider__c = 'AWS'
];
```

API Limits

API Limit Monitoring:

```
public class APILimitMonitor {

    public static void checkAndLogAPILimits() {
        Map<String, System.OrgLimit> limits = OrgLimits.getMap();
        System.OrgLimit apiLimit = limits.get('DailyApiRequests');

        Decimal usagePercent = (Decimal) apiLimit.getValue() / apiLimit.getLimit() * 100;

        if (usagePercent > 80) {
            sendAPILimitAlert(usagePercent);
        }

        // Log usage
        API_Usage_Log__c log = new API_Usage_Log__c(
            Date__c = Date.today(),
            API_Calls_Used__c = apiLimit.getValue(),
            Usage_Percentage__c = usagePercent
        );
        insert log;
    }

    public static Boolean canMakeAPICall() {
        Map<String, System.OrgLimit> limits = OrgLimits.getMap();
        Decimal usage = (Decimal) limits.get('DailyApiRequests').getValue() /
                        limits.get('DailyApiRequests').getLimit() * 100;
        return usage < 90; // Reserve 10% for critical ops
    }
}
```

OAuth & Authentication

OAuth Token Management:

```
public class OAuthAuthenticationService {

    public static String getAccessToken(String provider) {
        // Check cache first
        String cachedToken = Cache.Org.get('OAuth_Token_' + provider);
        if (cachedToken != null) return cachedToken;

        // Request new token
        HttpRequest req = new HttpRequest();
        req.setEndpoint('callout:' + provider + '_OAuth/token');
        req.setMethod('POST');
        req.setBody('grant_type=client_credentials');

        Http http = new Http();
        HttpResponse res = http.send(req);
        if (res.getStatusCode() == 200) {
            Map<String, Object> tokenData = (Map<String, Object>)
JSON.deserializeUntyped(res.getBody());
            String token = (String) tokenData.get('access_token');

            // Cache for 1 hour
            Cache.Org.put('OAuth_Token_' + provider, token, 3600);
            return token;
        }

        return null;
    }
}
```

Remote Site Settings

Remote Sites Configured:

- https://ec2.amazonaws.com (AWS EC2)
- https://management.azure.com (Azure)
- https://compute.googleapis.com (GCP)
- https://api.datadoghq.com (Datadog)
- https://api.newrelic.com (New Relic)

Phase 8: Data Management & Deployment

Goal - Implement comprehensive data management strategies and establish reliable deployment processes to ensure data integrity, system stability, and seamless migration between environments.

Data Import Wizard

Import Scenarios:

- **Cloud Providers**: Import AWS, Azure, GCP provider configurations
- **Cloud Resources**: Bulk import of existing cloud infrastructure inventory
- **Cost Records**: Import historical billing data from cloud providers
- **Budgets**: Import departmental budget allocations

Import Process:

1. Prepare CSV templates with required fields
2. Map CSV columns to Salesforce fields
3. Select import type (Insert, Update, Upsert)
4. Review and confirm import settings
5. Monitor import status and review errors

CSV Template Example:

```
Name,Type__c,Provider__c,Region__c,Monthly_Cost__c,Status__c

Production-Web-Server-01,Compute,AWS,us-west-2,250,Active

Production-Database-01,Database,Azure,eastus,500,Active
```

Data Loader

Use Cases:

- **Large Dataset Imports**: Import 10,000+ cloud resources
- **Data Extraction**: Export resource data for analysis
- **Bulk Updates**: Update multiple resource configurations
- **Data Migration**: Move data between orgs

Data Loader Operations:

- **Insert**: Add new cloud resources from CSV
- **Update**: Modify existing resource configurations
- **Upsert**: Insert or update based on External ID
- **Delete**: Remove decommissioned resources
- **Export**: Extract data for reporting and backup

Command Line Example:

```
# Export all cloud resources
dataloader.bat export


# Insert new resources
dataloader.bat insert Cloud_Resource__c resources.csv


# Upsert with External ID
dataloader.bat upsert Cloud_Resource__c resources.csv External_ID__c
```

Duplicate Rules

Cloud Resource Duplicate Rule:

- **Rule Name**: Prevent Duplicate Resources
- **Object**: Cloud_Resource__c
- **Matching Fields**: External_ID__c, Name + Provider__c
- **Action**: Alert user, Allow save with confirmation
- **Report**: Log duplicate attempts for audit

Cost Record Duplicate Rule:

- **Rule Name**: Prevent Duplicate Billing Entries
- **Object**: Cost_Record__c
- **Matching Fields**: Cloud_Resource__c + Billing_Period__c

- **Action**: Block save, Show error message
- **Report**: Track attempted duplicate cost entries

Matching Rules Configuration:

Cloud Resource Matching Rule:

- Match on: External_ID__c (Exact match)

- OR Name + Provider__c (Fuzzy match, 85% threshold)

- Applies to: Insert, Update operations

Data Export & Backup

Automated Backup Strategy:

- **Weekly Full Backup**: All cloud resources, cost records, budgets
- **Daily Incremental**: Changed records only
- **Monthly Archive**: Complete historical data snapshot

Export Schedules:

```
public class DataBackupScheduler implements Schedulable {
    public void execute(SchedulableContext sc) {
        // Export critical data
        List<Cloud_Resource__c> resources = [SELECT Id, Name, External_ID__c, Status__c FROM Cloud_Resource__c];
        List<Cost_Record__c> costs = [SELECT Id, Amount__c, Billing_Period__c FROM Cost_Record__c WHERE CreatedDate = LAST_N_DAYS:7];

        // Save to encrypted storage or external backup system
        BackupService.exportToSecureStorage(resources, costs);
    }
}

// Schedule weekly backups
```

```
System.schedule('Weekly Data Backup', '0 0 2 ? * SUN', new DataBackupScheduler());
```

Data Retention Policy:

- **Active Resources**: Retained indefinitely
- **Cost Records**: 7 years (compliance requirement)
- **Deployments**: 2 years
- **Error Logs**: 90 days
- **Archived Resources**: 3 years after decommission

Change Sets

Outbound Change Set (Dev to UAT):

- **Change Set Name**: CloudOps_Sprint1_Features
- **Components Included**:
  - Custom Objects (Cloud_Resource__c, Cost_Record__c)
  - Custom Fields (all new fields)
  - Page Layouts (updated layouts)
  - Flows (Budget Alert Flow, Resource Lifecycle Flow)
  - Apex Classes (CloudResourceManager, CostCalculationEngine)
  - Lightning Components (CloudResourceHealthGauge)
  - Reports and Dashboards

Deployment Steps:

1. Create outbound change set in source org
2. Add all modified components
3. Upload change set to target org
4. Review and validate components in target
5. Deploy during maintenance window
6. Run post-deployment tests
7. Notify users of new features

Change Set Best Practices:

- Deploy during off-peak hours (Sunday 2-4 AM)
- Include all dependent components
- Document changes in deployment notes
- Prepare rollback plan
- Test in sandbox before production

Unmanaged vs Managed Packages

CloudOps Central Package Strategy:

Unmanaged Package (Development/Custom Implementation):

- **Use Case**: Internal deployment, customizable solution
- **Components**: All custom objects, fields, code, UI
- **Advantages**: Fully customizable post-installation
- **Disadvantages**: No automatic updates, manual version management

Managed Package (Future Product Distribution):

- **Use Case**: App Exchange distribution, multi-org deployment
- **Components**: Core functionality with intellectual property protection
- **Advantages**: Version control, automatic updates, namespace protection
- **Disadvantages**: Limited customization, requires certification

Package Contents:

```
CloudOps Central v1.0 (Unmanaged)
 ├── Custom Objects (6)
|   ├── Cloud_Provider__c
|   ├── Cloud_Resource__c
|   ├── Cost_Record__c
|  └── ... (3 more)
 ├── Apex Classes (15)
 ├── Lightning Components (8)
 ├── Flows (12)
 ├── Reports (20)
└── Dashboards (5)
```

ANT Migration Tool

ANT Build Configuration (build.xml):

```
<project name="CloudOps Central" default="deploy" basedir="."
xmlns:sf="antlib:com.salesforce">
  <property file="build.properties"/>


  <target name="deploy">
    <sf:deploy username="${sf.username}"
              password="${sf.password}"
              serverurl="${sf.serverurl}"
              deployRoot="src"/>
  </target>


  <target name="retrieve">
    <sf:retrieve username="${sf.username}"
                password="${sf.password}"
                serverurl="${sf.serverurl}"
                retrieveTarget="src"
                unpackaged="package.xml"/>
  </target>
</project>
```

Package.xml Example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Cloud_Resource__c</members>
        <members>Cost_Record__c</members>
        <name>CustomObject</name>
    </types>
    <types>
        <members>CloudResourceManager</members>
        <members>CostCalculationEngine</members>
        <name>ApexClass</name>
    </types>
    <version>59.0</version>
</Package>
```

VS Code & SFDX

SFDX Project Setup:

```
# Initialize project
sfdx project:create --projectname cloudops-central

# Authorize org
sfdx auth:web:login --setalias cloudops-dev --instanceurl https://login.salesforce.com

# Create scratch org
sfdx org:create:scratch --definition-file config/project-scratch-def.json --alias cloudops-scratch --
set-default
```

```
# Push source to org
sfdx project:deploy:start --source-dir force-app


# Pull changes from org
sfdx project:retrieve:start --source-dir force-app
```

Project Structure:

```
cloudops-central/
 ├── force-app/
 |   └── main/
 |       └── default/
 |           ├── classes/
 |           |   ├── CloudResourceManager.cls
 |           |  └── CostCalculationEngine.cls
 |           ├── objects/
 |           |   ├── Cloud_Resource__c/
 |           |  └── Cost_Record__c/
 |           ├── lwc/
 |           |  └── cloudResourceHealthGauge/
 |           └── flows/
 |              └── Budget_Alert_Flow.flow-meta.xml
 ├── config/
 |   └── project-scratch-def.json
└── sfdx-project.json
```

SFDX Commands for Deployment:

```
# Deploy to production
sfdx project:deploy:start --target-org production
```

```
# Run tests

sfdx apex:test:run --code-coverage --result-format human


# Create package version

sfdx package:version:create --package "CloudOps Central" --installation-key-bypass


# Install package

sfdx package:install --package 04t... --target-org production
```

VS Code Extensions Required:

- Salesforce Extension Pack
- Salesforce CLI Integration
- Apex PMD (code quality)
- Prettier (code formatting)

Phase 9: Reporting, Dashboards & Security Review

Goal - Implement comprehensive reporting and analytics capabilities while ensuring robust security controls, audit compliance, and data protection across the entire cloud operations platform.

Reports (Tabular, Summary, Matrix, Joined)

Tabular Reports:

All Active Cloud Resources Report:

- **Type**: Tabular
- **Object**: Cloud Resources
- **Columns**: Name, Provider, Type, Region, Status, Monthly Cost, Owner
- **Filters**: Status equals "Active"

- **Sort**: Monthly Cost (descending)
- **Use Case**: Quick inventory view of all active infrastructure

Recent Deployments Report:

- **Type**: Tabular
- **Object**: Deployments
- **Columns**: Name, Environment, Status, Start Time, Duration, Responsible Engineer
- **Filters**: Created Date = Last 30 Days
- **Use Case**: Track deployment activities and success rates

Summary Reports:

Cost by Cloud Provider Summary:

- **Type**: Summary
- **Object**: Cost Records
- **Group By**: Cloud Provider
- **Summarize**: SUM(Amount), AVG(Amount), COUNT(Records)
- **Filters**: Billing Period = This Year
- **Charts**: Pie chart showing cost distribution
- **Use Case**: Executive overview of multi-cloud spending

Resource Utilization by Department:

- **Type**: Summary
- **Object**: Cloud Resources
- **Group By**: Department, Resource Type
- **Summarize**: COUNT(Resources), AVG(CPU_Utilization__c), SUM(Monthly_Cost__c)
- **Filters**: Status = Active
- **Use Case**: Department chargeback and capacity planning

Budget Performance Summary:

- **Type**: Summary

- **Object**: Budgets
- **Group By**: Department, Quarter
- **Summarize**: SUM(Allocated), SUM(Spent), Utilization %
- **Filters**: Year = This Year
- **Charts**: Bar chart comparing budget vs. actual
- **Use Case**: Financial planning and variance analysis

Matrix Reports:

Multi-Cloud Cost Matrix:

- **Type**: Matrix
- **Object**: Cost Records
- **Row Grouping**: Cloud Provider
- **Column Grouping**: Month (Billing Period)
- **Summarize**: SUM(Amount)
- **Use Case**: Trend analysis of spending across providers over time

Resource Type by Environment Matrix:

- **Type**: Matrix
- **Object**: Cloud Resources
- **Row Grouping**: Resource Type (Compute, Storage, Database, Network)
- **Column Grouping**: Environment (Production, Staging, Development)
- **Summarize**: COUNT(Resources), SUM(Monthly_Cost__c)
- **Use Case**: Infrastructure distribution and cost allocation

Joined Reports:

Complete Cloud Operations Report:

- **Type**: Joined
- **Blocks**:
    1. Cloud Resources (Active resources with utilization)

2. Cost Records (Monthly spending by resource)
3. Deployments (Recent deployment activities)

- **Common Field**: Cloud Resource ID
- **Use Case**: Comprehensive operational overview combining infrastructure, costs, and changes

Security and Compliance Dashboard Report:

- **Type**: Joined
- **Blocks**:
  1. Security Incidents (Open incidents by severity)
  2. Compliance Violations (Policy non-compliance items)
  3. Audit Trail (Recent configuration changes)
- **Use Case**: Security posture and compliance monitoring

Report Types

Custom Report Types Created:

Cloud Resources with Costs:

- **Primary Object**: Cloud Resources
- **Related Objects**: Cost Records (each resource may or may not have costs)
- **Fields Available**: All resource fields + cost amount, billing period, optimization opportunities
- **Use Case**: Analyze resource costs and identify optimization opportunities

Deployments with Affected Resources:

- **Primary Object**: Deployments
- **Related Objects**: Cloud Resources (through junction object)
- **Fields Available**: Deployment details + affected resource names, types, status changes
- **Use Case**: Impact analysis and deployment tracking

Budgets with Cost Allocation:

- **Primary Object**: Budgets

- **Related Objects**: Cost Records, Cloud Resources
- **Fields Available**: Budget details + actual spending + resource allocation
- **Use Case**: Budget variance analysis and forecasting

Dashboards

Executive Dashboard:

- **Component 1**: Total Monthly Cloud Spend (Metric)
- **Component 2**: Budget Utilization Gauge (85% threshold warning)
- **Component 3**: Multi-Cloud Cost Distribution (Pie Chart)
- **Component 4**: 12-Month Spending Trend (Line Chart)
- **Component 5**: Top 10 Most Expensive Resources (Horizontal Bar Chart)
- **Component 6**: Optimization Savings Opportunities (Metric with trend)
- **Component 7**: Active Resources Count by Provider (Donut Chart)
- **Component 8**: Recent Critical Alerts (Table)
- **Refresh**: Real-time (every 5 minutes)
- **Access**: Executive team, Finance, IT Management

Cloud Engineer Operations Dashboard:

- **Component 1**: Resource Health Score Distribution (Gauge Chart)
- **Component 2**: Resources Requiring Attention (Table - CPU >80% or Memory >85%)
- **Component 3**: Recent Deployments Status (Funnel Chart)
- **Component 4**: Performance Alerts by Severity (Stacked Bar Chart)
- **Component 5**: Resource Distribution by Region (Map Chart)
- **Component 6**: Average Response Time Trend (Line Chart)
- **Component 7**: Upcoming Maintenance Tasks (Table)
- **Component 8**: API Usage Statistics (Metric)
- **Refresh**: Every 15 minutes
- **Access**: Cloud Engineers, DevOps, SRE teams

FinOps Cost Management Dashboard:

- **Component 1**: Month-to-Date Spending (Metric with % change)
- **Component 2**: Budget vs Actual by Department (Grouped Bar Chart)
- **Component 3**: Cost Optimization Opportunities (Table with potential savings)
- **Component 4**: Reserved Instance Coverage (Gauge - target 75%)
- **Component 5**: Cost Anomaly Alerts (Table - spending spikes >20%)
- **Component 6**: Wasted Resources (idle/underutilized) (Table)
- **Component 7**: Cost Allocation by Service Type (Stacked Area Chart)
- **Component 8**: Forecast vs Actual Variance (Line Chart)
- **Refresh**: Daily at 6 AM
- **Access**: FinOps team, Finance, Budget owners

DevOps Deployment Dashboard:

- **Component 1**: Deployment Success Rate (Metric - target >95%)
- **Component 2**: Deployment Frequency by Environment (Bar Chart)
- **Component 3**: Mean Time to Deploy (Metric with 30-day trend)
- **Component 4**: Failed Deployments (Table with root cause)
- **Component 5**: Deployment Pipeline Status (Horizontal Stacked Bar)
- **Component 6**: Rollback Incidents (Metric)
- **Component 7**: Infrastructure Changes Timeline (Lightning Table)
- **Component 8**: Pending Approval Queue (Table)
- **Refresh**: Real-time
- **Access**: DevOps Engineers, Release Managers

Dynamic Dashboards

Multi-User Dynamic Dashboard Configuration:

- **Dashboard Name**: My Cloud Resources Dashboard
- **Running User**: Each user sees their own data

- **Dynamic Filters**: Department, Environment, Provider
- **Components Adjust Based On**: User's role, department, resource ownership
- **Use Case**: Personalized view of cloud resources and costs

Implementation:

```
// Dashboard filter logic
public class DynamicDashboardController {
  @AuraEnabled
  public static List<Cloud_Resource__c> getMyResources() {
    User currentUser = [SELECT Department FROM User WHERE Id = :UserInfo.getUserId()];

    return [
      SELECT Name, Type__c, Status__c, Monthly_Cost__c
      FROM Cloud_Resource__c
      WHERE Department__c = :currentUser.Department
      AND Status__c = 'Active'
    ];
  }
}
```

Sharing Settings

Organization-Wide Defaults (OWD):

- **Cloud Provider**: Public Read Only (master data visible to all)
- **Cloud Account**: Private (controlled by role hierarchy)
- **Cloud Resource**: Private (sensitive infrastructure data)
- **Cost Record**: Private (financial data restricted)
- **Budget**: Private (budget information controlled)
- **Deployment**: Public Read Only (collaboration visibility)

Sharing Rules:

Cloud Resources - DevOps Team:

- **Rule Type**: Criteria-based
- **Criteria**: Environment__c equals "Production"
- **Share With**: DevOps Team (Public Group)
- **Access Level**: Read/Write
- **Purpose**: Allow DevOps full access to production resources

Cost Records - Finance Department:

- **Rule Type**: Owner-based
- **Owner**: Any User
- **Share With**: Finance Department (Role)
- **Access Level**: Read Only
- **Purpose**: Enable financial oversight and audit

Budgets - Department Managers:

- **Rule Type**: Criteria-based
- **Criteria**: Department__c equals Current User's Department
- **Share With**: Department Manager (Role)
- **Access Level**: Read/Write
- **Purpose**: Allow managers to view and modify their department budgets

Field Level Security

Restricted Fields Configuration:

Cloud Resource Object:

- **External_ID__c**: Visible to Admins, Cloud Engineers only
- **API_Key__c**: Visible to Admins only (sensitive credentials)
- **Actual_Cost__c**: Read-only for Engineers, Read/Write for FinOps

Cost Record Object:

- **Discount_Applied__c**: Visible to FinOps and Finance only
- **Vendor_Contract_Terms__c**: Visible to Admins and Procurement only
- **Internal_Notes__c**: Visible to Finance team only

Budget Object:

- **Approved_Amount__c**: Read-only for all except Finance Directors
- **Emergency_Override__c**: Visible to Executives only

Session Settings

Session Security Configuration:

- **Session Timeout**: 2 hours for general users, 8 hours for administrators
- **Lock Sessions to IP**: Enabled for production org
- **Force Logout on Session Timeout**: Enabled
- **Require HTTPS**: Enforced for all connections
- **Enable Caching**: Enabled for performance (with secure cache only)
- **Clickjack Protection**: Enabled (deny framing by other sites)

Advanced Session Settings:

- Enable secure and persistent browser caching: Checked

- Require HttpOnly attribute: Enabled

- Use POST requests for cross-domain sessions: Enabled

- Enable Content Sniffing Protection: Enabled

- Enable XSS Protection: Enabled

Login IP Ranges

IP Restrictions by Profile:

Cloud Admin Profile:

- **Corporate Network**: 203.0.113.0 - 203.0.113.255
- **VPN Range**: 198.51.100.0 - 198.51.100.255
- **Emergency Access**: Bypass with MFA verification

Cloud Engineer Profile:

- **Office Network**: 203.0.113.0 - 203.0.113.255
- **Remote VPN**: 198.51.100.0 - 198.51.100.255
- **Mobile Access**: Requires additional verification

Manager Profile:

- **Any IP**: Allowed with MFA
- **Untrusted Networks**: Additional security challenge

API Integration User:

- **AWS Data Center**: 54.0.0.0 - 54.255.255.255
- **Azure Data Center**: 13.64.0.0 - 13.107.255.255
- **GCP Data Center**: 34.64.0.0 - 34.127.255.255

Audit Trail

Setup Audit Trail Monitoring:

- **Retention Period**: 180 days (6 months)
- **Monitored Actions**:
  - Security setting changes
  - User permission modifications
  - Sharing rule updates
  - Field-level security changes
  - Profile and role modifications
  - API access grants

Critical Changes Alert:

```
// Automated monitoring for critical changes
public class AuditTrailMonitor {


    public static void checkCriticalChanges() {
        List<SetupAuditTrail> recentChanges = [
            SELECT Action, CreatedBy.Name, CreatedDate, Section
            FROM SetupAuditTrail
            WHERE CreatedDate = TODAY
            AND Section IN ('Security', 'Sharing', 'Profiles')
        ];

        if (!recentChanges.isEmpty()) {
            sendSecurityAlert(recentChanges);
        }
    }
}
```

Field History Tracking:

Cloud Resource Object (Fields Tracked):

- Status__c (track all changes)
- Monthly_Cost__c (track significant changes)
- Owner (track ownership transfers)
- Environment__c (track production moves)

Cost Record Object (Fields Tracked):

- Amount__c (track all modifications)
- Billing_Period__c (detect backdating)
- Approval_Status__c (audit trail for approvals)

Budget Object (Fields Tracked):

- Amount__c (track budget modifications)
- Utilization_Percentage__c (monitor threshold breaches)
- Approval_Status__c (approval workflow tracking)

Security Best Practices Implemented

Data Protection:

- **Encryption at Rest**: Platform Encryption enabled for sensitive fields
- **Encryption in Transit**: HTTPS enforced for all communications
- **Data Masking**: Sensitive data masked in sandbox environments

Access Control:

- **Principle of Least Privilege**: Users granted minimum necessary permissions
- **Regular Access Reviews**: Quarterly review of user permissions
- **Deactivation Process**: Immediate access revocation for terminated employees

Monitoring and Compliance:

- **Login Forensics**: Monitor unusual login patterns
- **Event Monitoring**: Track API usage and data exports
- **Compliance Reports**: SOC2, ISO27001 audit-ready reports
- **Security Health Check**: Monthly automated security assessments

Incident Response:

- **Security Incident Workflow**: Automated case creation for security events
- **Escalation Path**: Defined chain of command for critical incidents
- **Post-Incident Review**: Documented lessons learned and remediation