Contents

Case API Configuration Analysis Report

Executive Summary

The Case API configuration folder at `/Users/agustin.fusaro/go/src/github.com/DataDog/dd-source/domains` implements a sophisticated multi-datacenter deployment system using Helm charts, Kubernetes templates, and Consul Template for dynamic configuration generation. The system supports deployment across 8+ global datacenters with environment-specific configurations for compliance, performance, and data locality.

Folder Structure and Contents

Main Configuration Directory: `config/k8s/`

```
config/k8s/
+-- BUILD.bazel                    # Bazel build configuration
+-- CHANGELOG.md                   # Configuration change history
+-- Chart.yaml                     # Helm chart metadata
+-- service.datadog.yaml           # Datadog service catalog definition
+-- values.yaml                    # Base Helm values
+-- tilt.star                      # Tilt development configuration
+-- ci/
|   +-- test-values.yaml           # CI/CD test values
+-- fabric/
|   +-- BUILD.bazel                # Fabric service mesh build config
|   +-- destination.yaml           # Fabric destination configuration
+-- templates/
|   +-- _case-api.ini.tpl          # Application configuration template
|   +-- configmap.yaml             # Kubernetes ConfigMap template
|   +-- deployment.yaml            # Kubernetes Deployment template
|   +-- fabric.yaml                # Fabric proxy configuration
|   +-- migrations-job.yaml        # Database migrations job
|   +-- service.yaml               # Kubernetes Service template
|   +-- serviceaccount.yaml        # Service account configuration
+-- values/
    +-- profiles/
        +-- devenv.yaml            # Development environment values
```

Configuration Architecture

1. Multi-Tier Configuration System

The configuration system operates on multiple levels:

1. **Base Configuration** (`values.yaml`): Common settings across all environments
2. **Datacenter-Specific Configuration**: Located in `../config/k8s/values/datacenters/`
3. **Tenant-Specific Configuration**: Located in `../config/k8s/values/tenants/case-management/datacer`
4. **Environment Profiles**: Development, staging, and production overrides

2. Helm Chart Structure

Chart.yaml defines the service as: - **Name**: `case-api` - **Version**: `0.0.6` - **Dependencies**: `helpers` chart from `@datadog` repository - **Description**: A Helm chart for Case API on Kubernetes

3. Deployment Configuration

The **deployment.yaml** template implements: - **Rolling updates** with zero downtime (maxUnavailable: 0, maxSurge: 1) - **Health checks** using gRPC health probes (with ARM64 support detection) - **Resource management** with configurable CPU/memory limits - **Security** with dedicated service accounts - **Observability** with Datadog agent integration

Datacenter-Specific Deployment

Supported Datacenters

The system supports deployment to 8 global datacenters:

Datacenter	Environment	Datadog Site	Special Features
us1.prod	Production	app.datadoghq.com	Primary US region
eu1.prod	Production	app.datadoghq.eu	EU data residency
us3.prod	Production	us3.datadoghq.com	Secondary US region
us5.prod	Production	us5.datadoghq.com	Tertiary US region
ap1.prod	Production	ap1.datadoghq.com	Asia Pacific 1
ap2.prod	Production	ap2.datadoghq.com	Asia Pacific 2
us1.fed	Federal	app.ddog-gov.com	FIPS compliance
us1.staging	Staging	dd.datad0g.com	Testing environment

Conditional Environment Variables

The system conditionally sets environment variables based on deployment context through the `case-management.env_variables` template:

```
# Template Definition (from _labels.tpl)
{{- define "case-management.env_variables" -}}
- name: GODEBUG
  value: "netdns=cgo"
- name: DD_DATACENTER
  value: '{{ $.Values.global.datacenter.datacenter }}'
- name: DD_SITE
  value: '{{ $.Values.global.dd_site }}'
- name: STATSD_URL
  value: unix:///var/run/datadog-agent/statsd.sock
- name: TENANT_NAME
  value: '{{ $.Values.tenant_name }}'
{{- end -}}
```

Datacenter-Specific Configuration Examples

US1 Production

```yaml
# From us1.prod.dog.yaml
dd_site: "https://app.datadoghq.com"
datacenter:
  datacenter: "us1"
  arm_enabled: false
```

EU1 Production

```yaml
# From eu1.prod.dog.yaml
dd_site: "https://app.datadoghq.eu"
datacenter:
  datacenter: "eu1"
  arm_enabled: false
```

US1 Federal (Government)

```yaml
# From us1.fed.dog.yaml
dd_site: "https://app.ddog-gov.com"
dd_partition: "gov-us"
datacenter:
  datacenter: "us1"
  arm_enabled: false
```

Infrastructure Components

1. Database Configuration

Each datacenter maintains isolated database clusters:

US1 Production:

```yaml
db:
  master:
    host: case-management-rds.postgres.us1.prod.dog
    port: 5433
  replicas:
    host: case-management-rds.postgres.us1.prod.dog
    port: 5433
```

EU1 Production:

```yaml
db:
  master:
    host: case-management-db.postgres.eu1.prod.dog
    port: 5432
  replicas:
    host: case-management-db-ro.postgres.eu1.prod.dog
    port: 5432
```

US1 Federal (with Fabric proxy):

```yaml
db:
  master:
    host: case-management-rds.postgres.us1.fed.fabric.dog
    port: 5432
  replicas:
    host: case-management-rds-ro.postgres.us1.fed.fabric.dog
    port: 5432
```

2. Kafka Configuration

Datacenter-isolated Kafka clusters with unique identifiers:

Datacenter	Cluster ID	Bootstrap Servers
us1.prod	kafka-case-management-56de	kafka-case-management-56de.us1.prod.dog:9092
eu1.prod	kafka-case-management-ad55	kafka-case-management-ad55.eu1.prod.dog:9092
us3.prod	kafka-case-management-61d9	kafka-case-management-61d9.us3.prod.dog:9092
us5.prod	kafka-case-management-0e5e	kafka-case-management-0e5e.us5.prod.dog:9092
ap1.prod	kafka-case-management-6609	kafka-case-management-6609.ap1.prod.dog:9092
ap2.prod	kafka-case-management-ca9c	kafka-case-management-ca9c.ap2.prod.dog:9092
us1.fed	kafka-case-management-c096	kafka-case-management-c096.us1.fed.dog:9092
us1.staging	kafka-case-management-c676	kafka-case-management-c676.us1.staging.dog:9092

3. Container Image Selection

The deployment template conditionally selects container images based on compliance requirements:

```
# Standard deployment
{{- if eq $.Values.global.dd_partition "gov-us" }}
image: "{{ .Values.global.oci.registry }}/{{ .Values.cnab.images.caseapifips.repository }}@{{
{{- else }}
image: "{{ .Values.global.oci.registry }}/{{ .Values.cnab.images.caseapi.repository }}@{{ .Valu
{{- end }}
```

- **Standard environments**: Uses `caseapi` image
- **Government environments**: Uses `caseapifips` image (FIPS-compliant)

Configuration Generation Process

1. Consul Template Integration

The system uses Consul Template for dynamic configuration generation:

Init Container (from deployment.yaml:76-98):

```
initContainers:
- name: consul-template-init
  image: "{{ $.Values.global.docker.registry }}/{{ $.Values.global.consulTemplate.image.name }}
  volumeMounts:
  - name: config
    mountPath: /etc/datadog
  - name: templates
    mountPath: /etc/templates
  args:
  - -once
  - -template=/etc/templates/case-api.ini:/etc/datadog/case-api.ini
  env:
  - name: DD_DATACENTER
    value: '{{ $.Values.global.datacenter.datacenter }}'
  - name: DD_SITE
    value: '{{ $.Values.global.dd_site }}'
```

2. Configuration Template

The `_case-api.ini.tpl` template includes:

```
{{ include "case-management.postgres" $ }}      # Database configuration
{{ include "case-management.consul_config" $ }} # Consul service discovery
{{ include "case-management.features_flag" $ }} # Feature flags
{{ include "case-management.dd_clients" $ }}    # Datadog client configuration
{{ include "case-management.elasticsearch" $ }} # Elasticsearch configuration
```

Service Discovery and Networking

1. Fabric Service Mesh

The service integrates with Datadog's Fabric service mesh:

Annotations (from deployment.yaml:33-36):

```
annotations:
  beta.fabric.datadoghq.com/inject-lifecycle: "inject"
  beta.fabric.datadoghq.com/egress-gateway: "true"
  beta.fabric.datadoghq.com/proxy-configmap: "{{ $.Release.Name }}-fabric-proxy-config"
```

Destination Configuration (from fabric/destination.yaml):

```
spec:
  defaultConnectionConfig:
    protocol: HTTP2
    tls: {}
  endpointGroups:
    main:
      selector:
        kubernetesServiceSelector:
```

```
          name: {{ $.Release.Name }}
          namespace: {{ $.Release.Namespace }}
          port: 6481
```

2. Health Checks

Architecture-aware health checks:

```
{{- if $.Values.global.datacenter.arm_enabled }}
exec:
  command: ["/grpc_health_probe_arm64", "--addr=127.0.0.1:6481", "--tls=true"]
{{- else }}
exec:
  command: ["/grpc_health_probe", "--addr=127.0.0.1:6481", "--tls=true"]
{{- end }}
```

Deployment Automation

1. CNAB Workflows (from BUILD.bazel)

```
DATACENTERS_PER_TENANT = {
    CASE_MANAGEMENT_TENANT_NAME: ALL_DATACENTERS_WITH_FED,  # Includes federal
    ON_CALL_TENANT_NAME: ALL_DATACENTERS,                   # Standard only
}

casem_cnab_workflows(
    app_name = APP_NAME,
    app_path = APP_PATH,
    installations_per_tenant = DATACENTERS_PER_TENANT,
    with_fabric_destinations = True,
)
```

2. Automated Deployment Targets (from service.datadog.yaml)

Conductor Configuration:

```
conductor:
  targets:
    - name: staging
      schedule: "0 5 * * Mon"          # Monday 5 AM
      branch: casem/staging
      slack: "case-management-releases-stg"
      ci_bazel_targets:
        - "//domains/case_management/apps/case-api/config/k8s:staging.publish"

    - name: prod
      schedule: "0 9 * * Mon"          # Monday 9 AM
      slack: "case-management-releases"
      ci_bazel_targets:
        - "//domains/case_management/apps/case-api/config/k8s:prod.publish"
```

```
      - "//domains/case_management/apps/case-api/config/k8s:prod-fast.publish"

  - name: gov
    schedule: "0 9 * * Mon"          # Monday 9 AM
    slack: "case-management-releases"
    artifacts:
      - "domains/case_management/apps/case-api/config/k8s/gov"
```

Security and Compliance

1. FIPS Compliance

Government deployments implement FIPS compliance through: - **Dedicated FIPS container images** (`caseapifips`) - **Specialized Datadog site** (`app.ddog-gov.com`) - **Separate datacenter partition** (`dd_partition: "gov-us"`) - **Isolated infrastructure** (separate Kafka, databases, etc.)

2. Secret Management

Secrets are managed through Consul's KV store:

```
user: '{% with secret "kv/k8s/{{ $.Release.Namespace }}/{{ $.Release.Name }}/postgres/migratio
password: '{% with secret "kv/k8s/{{ $.Release.Namespace }}/{{ $.Release.Name }}/postgres/migra
```

3. Network Security

- **TLS termination** at ingress level
- **mTLS** between services via Fabric mesh
- **Network policies** for service isolation
- **Service accounts** with minimal privileges

Development and Testing

1. Development Environment (devenv.yaml)

```
arm_enabled: false
replicas: 1
useTilt: true                # Tilt integration for local development

db:
  master:
    host: postgres           # Local PostgreSQL
    port: 5432
  name: dogdata
  user: dog
  password: dog

dd_clients:
  url: "https://dd.datad0g.com"  # Staging environment
```

2. Tilt Integration

The `tilt.star` file provides local development support with hot reloading and live updates.

3. CI/CD Testing

Test values in `ci/test-values.yaml` provide configuration for automated testing pipelines.

Key Configuration Patterns

1. Conditional Environment Variables

To conditionally set environment variables based on datacenter:

```
# Method 1: In Helm templates
{{- if eq $.Values.global.datacenter.datacenter "us1" }}
- name: CUSTOM_VAR
  value: "us1-specific-value"
{{- else if eq $.Values.global.datacenter.datacenter "eu1" }}
- name: CUSTOM_VAR
  value: "eu1-specific-value"
{{- end }}

# Method 2: Using datacenter-specific values files
# In us1.prod.dog.yaml:
custom_var: "us1-specific-value"
# In eu1.prod.dog.yaml:
custom_var: "eu1-specific-value"

# Then reference in template:
- name: CUSTOM_VAR
  value: '{{ $.Values.custom_var }}'
```

2. Government vs Commercial Deployments

```
{{- if eq $.Values.global.dd_partition "gov-us" }}
# Government-specific configuration
{{- else }}
# Commercial configuration
{{- end }}
```

3. Architecture-Specific Configurations

```
{{- if $.Values.global.datacenter.arm_enabled }}
# ARM64-specific configuration
{{- else }}
# x86_64 configuration
{{- end }}
```

Monitoring and Observability

1. Datadog Integration

Each datacenter connects to its region-specific Datadog instance: - **Metric collection** via StatsD unix socket - **Log aggregation** with structured logging (`log_format: dd-go-std`) - **APM tracing** with service tags - **Dashboard links** in service catalog

2. Health Monitoring

- **Readiness probes** with gRPC health checks
- **Service discovery** integration
- **Performance metrics** collection
- **Error tracking** and alerting

Troubleshooting Guide

Common Configuration Issues

1. **Database Connection Failures**
   - Check datacenter-specific database hosts in values files
   - Verify Consul Template secret resolution
   - Confirm network policies allow database access
2. **Kafka Connection Issues**
   - Validate cluster ID matches datacenter
   - Check bootstrap server endpoints
   - Verify topic permissions and configurations
3. **Environment Variable Problems**
   - Review `case-management.env_variables` template
   - Check datacenter-specific values files
   - Validate Helm value precedence
4. **FIPS Compliance Issues**
   - Ensure `dd_partition: "gov-us"` is set correctly
   - Verify FIPS container images are being used
   - Check government-specific endpoints

Recommendations

Best Practices for Adding New Environment Variables

1. **Add to the central template**: Update `case-management.env_variables` in `_labels.tpl`
2. **Use datacenter-specific values**: Define different values per datacenter in respective YAML files
3. **Test in staging first**: Deploy to `us1.staging` before production datacenters
4. **Consider compliance**: Separate configurations for government vs commercial deployments

Configuration Management

1. **Version control**: All configuration changes should be tracked in git
2. **Change management**: Use the CHANGELOG.md to document configuration modifications
3. **Testing**: Leverage CI/CD pipelines with test-values.yaml

4. **Rollback planning**: Maintain configuration rollback procedures

---

Report Generated: `date`
Configuration Version: 0.0.6
Helm Chart: case-api
Team: case-management