Iteration 3.2: Helm Templating Deep Dive

Duration: ~1.5-2 hours
Difficulty: ⭐⭐⭐ Intermediate-Advanced

🎯 Learning Objectives

- ✅ Master Go template syntax
- ✅ Use built-in objects (.Values, .Release, .Chart, .Template)
- ✅ Apply functions and pipelines
- ✅ Implement conditionals (if/else)
- ✅ Use loops (range)
- ✅ Create template helpers (_helpers.tpl)
- ✅ Debug template rendering

📚 Template Fundamentals

Built-in Objects

```
{{ .Values.key }}           # From values.yaml
{{ .Release.Name }}         # Release name
{{ .Release.Namespace }}    # Release namespace
{{ .Chart.Name }}           # Chart name
{{ .Chart.Version }}        # Chart version
{{ .Template.Name }}        # Current template name
```

Functions and Pipelines

```
# Functions
{{ default "defaultValue" .Values.optional }}
{{ quote .Values.string }}
{{ upper .Values.name }}
{{ lower .Values.name }}

# Pipelines (similar to Unix pipes)
{{ .Values.name | upper | quote }}
{{ .Values.number | toString }}
```

Conditionals

```
{{- if .Values.enabled }}
apiVersion: v1
kind: Service
# ... service definition
```

```
{{- else }}
# Alternative
{{- end }}

# Multiple conditions
{{- if and .Values.enabled .Values.production }}
# ...
{{- else if .Values.development }}
# ...
{{- else }}
# ...
{{- end }}
```

Loops

```
# Range over list
{{- range .Values.environments }}
- name: {{ . }}
{{- end }}

# Range with index
{{- range $index, $value := .Values.items }}
  {{ $index }}: {{ $value }}
{{- end }}

# Range over map
{{- range $key, $value := .Values.config }}
  {{ $key }}: {{ $value }}
{{- end }}
```

🛠️ Hands-On Examples

Example 1: Conditional Service Type

values.yaml:

```
service:
  enabled: true
  type: ClusterIP  # or NodePort, LoadBalancer
  nodePort: 30080  # only used if type is NodePort
```

templates/service.yaml:

```
{{- if .Values.service.enabled }}
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.Name }}-service
spec:
  type: {{ .Values.service.type }}
  ports:
  - port: 80
    targetPort: 8080
    {{- if eq .Values.service.type "NodePort" }}
    nodePort: {{ .Values.service.nodePort }}
    {{- end }}
  selector:
    app: {{ .Release.Name }}
{{- end }}
```

Example 2: Environment Variables from Map

values.yaml:

```
env:
  LOG_LEVEL: info
  DEBUG: "false"
  MAX_CONNECTIONS: "100"
```

templates/deployment.yaml:

```
env:
{{- range $key, $value := .Values.env }}
- name: {{ $key }}
  value: {{ $value | quote }}
{{- end }}
```

Example 3: Multiple Replicas Based on Environment

values.yaml:

```
environment: production  # or development

replicas:
  development: 1
  staging: 2
  production: 3
```

templates/deployment.yaml:

```
spec:
  replicas: {{ index .Values.replicas .Values.environment }}
```

Example 4: Template Helpers

templates/_helpers.tpl:

```
{{/*
Common labels
*/}}
{{- define "myapp.labels" -}}
app: {{ .Release.Name }}
chart: {{ .Chart.Name }}-{{ .Chart.Version }}
release: {{ .Release.Name }}
heritage: {{ .Release.Service }}
{{- end }}

{{/*
Selector labels
*/}}
{{- define "myapp.selectorLabels" -}}
app: {{ .Release.Name }}
release: {{ .Release.Name }}
{{- end }}

{{/*
Full name
*/}}
{{- define "myapp.fullname" -}}
{{- if .Values.fullnameOverride }}
{{- .Values.fullnameOverride | trunc 63 | trimSuffix "-" }}
{{- else }}
{{- printf "%s-%s" .Release.Name .Chart.Name | trunc 63 | trimSuffix "-"
}}
{{- end }}
{{- end }}
```

Usage in templates/deployment.yaml:

```
metadata:
  name: {{ include "myapp.fullname" . }}
  labels:
    {{- include "myapp.labels" . | nindent 4 }}
spec:
  selector:
    matchLabels:
      {{- include "myapp.selectorLabels" . | nindent 6 }}
  template:
    metadata:
      labels:
        {{- include "myapp.labels" . | nindent 8 }}
```

Example 5: toYaml and Indentation

values.yaml:

```yaml
resources:
  requests:
    memory: "128Mi"
    cpu: "100m"
  limits:
    memory: "256Mi"
    cpu: "200m"
```

templates/deployment.yaml:

```yaml
resources:
{{ toYaml .Values.resources | indent 10 }}

# Or with nindent (newline + indent):
resources:
  {{- toYaml .Values.resources | nindent 10 }}
```

🎯 Practice Exercises

Exercise 1: Multi-Environment ConfigMap

Create a chart that generates different ConfigMaps based on environment:

values.yaml:

```yaml
environment: production

config:
  development:
    LOG_LEVEL: debug
    CACHE: "false"
  production:
    LOG_LEVEL: info
    CACHE: "true"
```

templates/configmap.yaml:

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-config
```

```
data:
  {{- range $key, $value := (index .Values.config .Values.environment) }}
  {{ $key }}: {{ $value | quote }}
  {{- end }}
```

Exercise 2: Conditional Ingress

Create Ingress only if enabled:

values.yaml:

```
ingress:
  enabled: false
  host: example.com
  tls:
    enabled: false
    secretName: tls-secret
```

templates/ingress.yaml:

```
{{- if .Values.ingress.enabled }}
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: {{ .Release.Name }}-ingress
spec:
  {{- if .Values.ingress.tls.enabled }}
  tls:
  - hosts:
    - {{ .Values.ingress.host }}
    secretName: {{ .Values.ingress.tls.secretName }}
  {{- end }}
  rules:
  - host: {{ .Values.ingress.host }}
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: {{ .Release.Name }}-service
            port:
              number: 80
{{- end }}
```

Exercise 3: Dynamic Volume Mounts

values.yaml:

```
volumes:
  - name: config
    mountPath: /etc/config
    configMap: app-config
  - name: secrets
    mountPath: /etc/secrets
    secret: app-secrets
```

templates/deployment.yaml:

```
volumeMounts:
{{- range .Values.volumes }}
- name: {{ .name }}
  mountPath: {{ .mountPath }}
{{- end }}

volumes:
{{- range .Values.volumes }}
- name: {{ .name }}
  {{- if .configMap }}
  configMap:
    name: {{ .configMap }}
  {{- else if .secret }}
  secret:
    secretName: {{ .secret }}
  {{- end }}
{{- end }}
```

✅ Validation

```
# Template rendering
helm template my-release ./my-chart

# Debug mode
helm install --dry-run --debug my-release ./my-chart

# Test with different values
helm template my-release ./my-chart \
  --set environment=production

# Lint
helm lint ./my-chart
```

📚 Key Functions

String Functions:

```
{{ upper "hello" }}           # HELLO
{{ lower "HELLO" }}           # hello
{{ title "hello world" }}     # Hello World
{{ trim "  hello  " }}        # hello
{{ quote "value" }}           # "value"
{{ replace " " "-" "hello world" }}  # hello-world
```

Math Functions:

```
{{ add 1 2 }}                 # 3
{{ sub 5 2 }}                 # 3
{{ mul 3 4 }}                 # 12
{{ div 10 2 }}                # 5
```

Type Conversion:

```
{{ toString 123 }}            # "123"
{{ toYaml .Values.obj }}      # YAML format
{{ toJson .Values.obj }}      # JSON format
```

Logic Functions:

```
{{ and .Values.a .Values.b }}
{{ or .Values.a .Values.b }}
{{ not .Values.a }}
{{ eq .Values.a "value" }}
{{ ne .Values.a "value" }}
```

🎯 Next Steps

Next: Iteration 3.3: Values and Overrides

Learn advanced value management and environment-specific configurations.

See `./solution/` for complete examples.