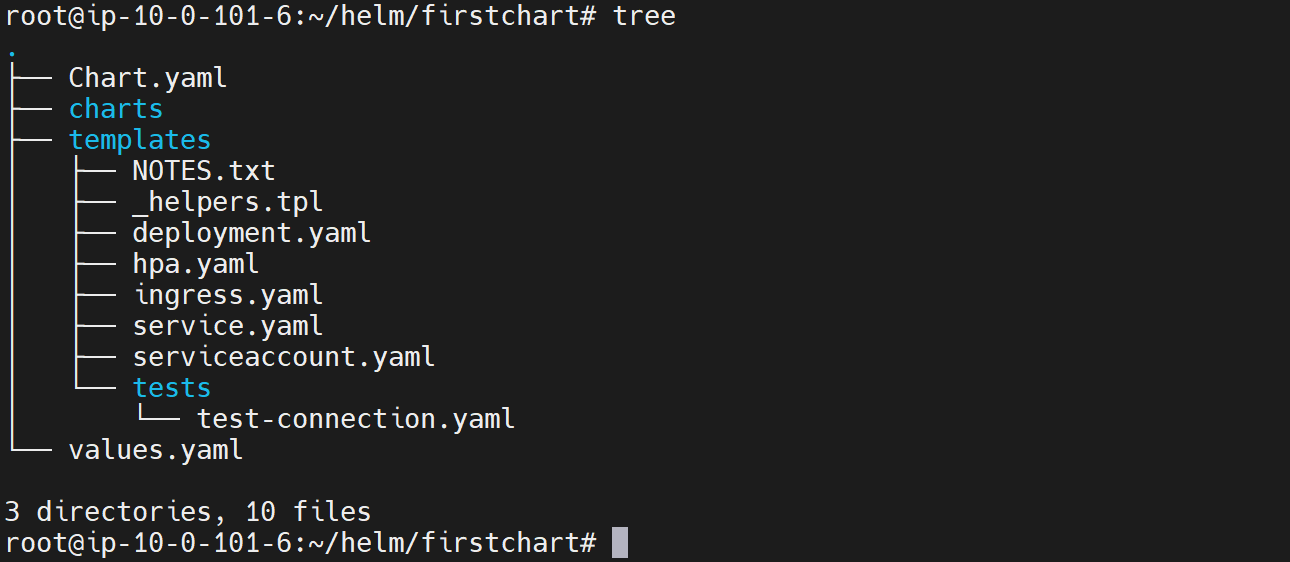
**37. Templates in brief**

--- we will talk about the Templates folder, which is the heart of every helm chart. It has the templates of yaml files using which the Kubernetes manifest will be generated and when those manifests are sent to kubernetes. It will create all the kubernetes resources for our application

--- tree



**Deployment.yml**

--- Start with the deployment dot yaml. All these files are responsible for generating the final manifest that will be sent to Kubernetes.

--- So, this deployment.yml looks just like a deployment yaml file, which we use directly to create Kubernetes deployment.

--- **deployment.yml**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: {{ include "firstchart.fullname" . }}

  labels:

    {{- include "firstchart.labels" . | nindent 4 }}

spec:

  {{- if not .Values.autoscaling.enabled }}

  replicas: {{ .Values.replicaCount }}

  {{- end }}

  selector:

    matchLabels:

      {{- include "firstchart.selectorLabels" . | nindent 6 }}

  template:

    metadata:

      {{- with .Values.podAnnotations }}

      annotations:

        {{- toYaml . | nindent 8 }}

      {{- end }}

      labels:

        {{- include "firstchart.selectorLabels" . | nindent 8 }}

    spec:

      {{- with .Values.imagePullSecrets }}

      imagePullSecrets:

        {{- toYaml . | nindent 8 }}

      {{- end }}

      serviceAccountName: {{ include "firstchart.serviceAccountName" . }}

      securityContext:

        {{- toYaml .Values.podSecurityContext | nindent 8 }}

      containers:

        - name: {{ .Chart.Name }}

          securityContext:

            {{- toYaml .Values.securityContext | nindent 12 }}

          image: "{{ .Values.image.repository }}:{{ .Values.image.tag | default .Chart.AppVersion }}"

          imagePullPolicy: {{ .Values.image.pullPolicy }}

          ports:

            - name: http

              containerPort: 80

              protocol: TCP

          livenessProbe:

            httpGet:

              path: /

              port: http

          readinessProbe:

            httpGet:

              path: /

              port: http

          resources:

            {{- toYaml .Values.resources | nindent 12 }}

      {{- with .Values.nodeSelector }}

      nodeSelector:

        {{- toYaml . | nindent 8 }}

      {{- end }}

      {{- with .Values.affinity }}

      affinity:

        {{- toYaml . | nindent 8 }}

      {{- end }}

      {{- with .Values.tolerations }}

      tolerations:

        {{- toYaml . | nindent 8 }}

      {{- end }}

--- **note** - it has placeholder in this template. This is the Google Go templating syntax. An entire section is dedicated to this where you will master This syntax, which is the next section. So, this is a brief introduction to templates.

--- **Note** - If you want, you can delete some of these templates. These were generated when you created the chart but if you think you are project does not need a service account, you can just get rid of serviceaccount.yml file.

--- If you think your project does not need an ingress configuration, get rid of that yml file so you can freely delete, modify, add. If you need some other type of resource here, pod.yml file, etc. you can just add it here and automatically. Helm will take that template, generate a manifest and send it to kubernetes That's the beauty of Helm.

**hpa.yml**

--- Let us look at one more file hpa.yml. Again, this looks just like a file that we use to create horizontal pod auto scalar type of resource on Kubernetes, but it has conditional logic.

--- cat hpa.yml

{{- if .Values.autoscaling.enabled }}

apiVersion: autoscaling/v2beta1

kind: HorizontalPodAutoscaler

metadata:

  name: {{ include "firstchart.fullname" . }}

  labels:

    {{- include "firstchart.labels" . | nindent 4 }}

spec:

  scaleTargetRef:

    apiVersion: apps/v1

    kind: Deployment

    name: {{ include "firstchart.fullname" . }}

  minReplicas: {{ .Values.autoscaling.minReplicas }}

  maxReplicas: {{ .Values.autoscaling.maxReplicas }}

  metrics:

    {{- if .Values.autoscaling.targetCPUUtilizationPercentage }}

    - type: Resource

      resource:

        name: cpu

        targetAverageUtilization: {{ .Values.autoscaling.targetCPUUtilizationPercentage }}

    {{- end }}

    {{- if .Values.autoscaling.targetMemoryUtilizationPercentage }}

    - type: Resource

      resource:

        name: memory

        targetAverageUtilization: {{ .Values.autoscaling.targetMemoryUtilizationPercentage }}

    {{- end }}

{{- end }}

--- **note** – the same goes to the other projects as well.