One-Day Helm for Beginners

Duration: 1 Day (6–8 hours)
Description: Fast-paced, hands-on introduction to Helm for deploying and managing Kubernetes applications using charts.
Learning outcomes: - Explain what Helm is and why to use it.
- Install Helm, kubectl, and Minikube and validate the toolchain.
- Discover and use Helm charts from repositories.
- Create a simple chart with templates and values.
- Install, upgrade, rollback, and uninstall releases.
- Work with repositories, namespaces, values, and overrides.
- Lint, test, and troubleshoot basic Helm issues.
- Deploy a small web app via a Helm chart. - Create reproducible builds of Kubernetes applications using charts and values.
- Share applications as Helm charts and run third-party charts safely.
- Manage releases of Helm packages across namespaces and revisions.

Format of the course: - Interactive lecture and discussion.
- Lots of exercises and practice.
- Hands-on implementation in a live-lab environment.

Table of Contents

Table of Contents - Day Plan & Timing - Environment Setup (08:30–09:00) - Installation: kubectl, Minikube, Helm - Module 1 (09:00–10:00): Helm Fundamentals - Module 2 (10:00–11:00): Installing & Using Helm - Module 3 (11:00–12:00): Creating Helm Charts - Lunch

Environment Setup (08:30–09:00)

Objectives: Ensure attendees have kubectl, Minikube, Helm installed and working locally.

Steps (macOS/Linux): 1) Install kubectl (latest stable):

```
curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/$(una
chmod +x kubectl && sudo mv kubectl /usr/local/bin/
kubectl version --client
```

Expected: `Client Version: vX.Y.Z`.

2) Install Minikube:

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-$(uname | tr '[:uppe
sudo install minikube-$(uname | tr '[:upper:]' '[:lower:]')-amd64 /usr/local/bin/minikube
minikube start --driver=docker
kubectl get nodes
```

Expected: single node `Ready`.

3) Install Helm:

```
curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash
helm version
```

Expected: `version.BuildInfo{Version:"v3.x.x"}`.

Troubleshooting: - If Docker not running: start Docker Desktop or switch driver with `--driver=hyperkit` or `--driver=virtualbox` if available.
- If kubectl cannot reach cluster: `minikube status`; if stopped, run `minikube start`.
- PATH issues: ensure `/usr/local/bin` precedes other paths.

Installation Section (detailed)

Installation Section (detailed)

Module 1 – Helm Fundamentals (09:00–10:00)

Learning objectives:

- Define Helm, charts, releases, and repositories.
- Understand chart structure at a high

Installation Section (detailed)

Plain-language explanation: Helm is the package manager for Kubernetes. A **chart** is the package. A **release** is an installed instance of a chart in a cluster. A **repository** stores charts.

Installation Section (detailed)

Diagram (text): Think of Helm as `apt/yum` for Kubernetes. Boxes: “Helm CLI” -> “Chart Repo” (pulls chart) -> “Kubernetes Cluster” (installs as Release). Release connects to objects (Deployments, Services, ConfigMaps)

Installation Section (detailed)

Step-by-step demo (read-only):

Installation Section (detailed)

Extra quick checks:
- Compare `helm search hub nginx` vs `helm search repo nginx` after adding repos to see the difference between Hub and local indexes.
- Run `helm plugin list` (should be empty) to highlight that plugins

Installation Section (detailed)

Minilab (15–20 min):

- Goal: Explore Helm help and inspect a chart README.
- Steps: run the commands above; note a chart name and version.
- Expected: list of charts, README content visible.

Installation Section (detailed)
Review questions: - What is the difference between a chart and a release? - Where do charts live? - Why use Helm instead of plain kubectl apply?

Module 2 – Installing & Using Helm (10:00–11:00)

Learning objectives: - Install Helm (already done) and configure a repo.
- Add, search, pull charts.
- Install first release into Minikube.

Plain-language explanation: Repositories act like registries. Adding a repo makes charts discoverable. Installing creates Kubernetes resources using chart templates and default values unless overridden.

Diagram (text): Steps pipeline: `helm repo add` -> `helm repo update` -> `helm search repo` -> `helm install demo-release chartname` -> Kubernetes objects created -> Service exposed.

Step-by-step demo:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
helm repo update
helm search repo nginx
helm pull bitnami/nginx --untar -d /tmp/helm-nginx
ls /tmp/helm-nginx/nginx
helm install demo-nginx bitnami/nginx
kubectl get pods,svc
helm get manifest demo-nginx | head -n 20
helm status demo-nginx
helm uninstall demo-nginx --keep-history
```

Extra examples: - Preview without changing the cluster: `helm install demo-nginx bitnami/nginx --dry-run --debug`. - Show default values: `helm show values bitnami/nginx | head -n 20` and note how they map to the README. - Reinstall with a different release name (e.g., `demo-nginx2`) to illustrate multiple releases of the same chart.

Mini-lab (20–30 min): - Install `bitnami/nginx` as `lab-nginx`.
- Verify pod and service.
- Port-forward: `kubectl port-forward svc/lab-nginx 8080:80` and curl localhost:8080.
- Cleanup: `helm uninstall lab-nginx`.

Review questions: - What does `helm repo update` do?
- How do you see what a chart will create before installing? (`helm show`, `helm template`)

Module 3 – Creating Helm Charts (11:00–12:00) **Learning objectives:** - Scaffold a new chart. - Understand key files: `Chart.yaml`, `values.yaml`, `templates/`. - Render templates locally with values.

Lunch Break (12:00–13:00)

Module 4 – Deploying & Managing Releases (13:00–14:00)

Learning objectives: - Install from local chart.
- Upgrade and roll back releases.
- Uninstall cleanly.

Plain-language explanation: Helm manages release history. Each `helm install/upgrade` increments a revision. `helm rollback` re-applies a previous revision. State lives in Kubernetes secrets/configmaps in the release namespace.

Diagram (text): Timeline: Release v1 (install) -> v2 (upgrade) -> rollback to v1. Helm stores revisions; Kubernetes objects change per revision.

Step-by-step demo:

```
helm install web-hello ./web-hello
kubectl get pods,svc
helm upgrade web-hello ./web-hello --set service.type=NodePort
helm history web-hello
helm rollback web-hello 1
helm uninstall web-hello
helm upgrade web-hello ./web-hello --set replicaCount=3 --atomic --timeout 2m
helm status web-hello
```

Extra examples: - Use `--atomic` on upgrade to auto-rollback on failure. - Set a custom namespace: `helm install web-hello ./web-hello -n demo --create-namespace` and compare `helm list -A` output. - Show how history persists after uninstall when `--keep-history` is used.

Mini-lab (20–30 min): - Install your `web-hello` chart.
- Change `replicaCount` from 1 to 2 via `--set replicaCount=2`.
- Verify pods scale.
- Roll back to revision 1.
- Uninstall at end.

Review questions: - Where does Helm store release history?
- What command shows revisions?
- How to undo a bad upgrade?

Module 5 – Repositories, Namespaces, Values, Overrides (14:00–15:00)

Learning objectives:

- Work with multiple repos and namespaces.
- Override values using CLI and files.
- Understand precedence of values.

Module 5 – Repositories,

Module 5 – Repositories, Namespaces, Values, Overrides (14:00–15:00)

Learning objectives:

- Work with multiple repos and namespaces.
- Override values using CLI and files.
- Understand precedence of values.

Plain-language explanation:

`helm`

Module 5 – Repositories, Namespaces, Values, Overrides (14:00–15:00)

Learning objectives:
- Work with multiple repos and namespaces.
- Override values using CLI and files.
- Understand precedence of values.

Diagram (text): Flow: `helm lint` -> `helm`

Module 5 – Repositories, Namespaces, Values, Overrides (14:00–15:00)

Learning objectives:
- Work with multiple repos and namespaces.
- Override values using CLI and files.
- Understand precedence of values.

Step-by-step demo:

Module 5 – Repositories, Namespaces, Values, Overrides (14:00–15:00) **Learning objectives:** - - Work with multiple repos and namespaces. - - Override values using CLI and files. - Understand precedence of values.

Common errors & **fixes:** - Image

Module 5 – Repositories, Namespaces, Values, Overrides (14:00–15:00) **Learning objectives:** - Work with multiple repos and namespaces. - Override values using CLI and files. - Understand precedence of values.

Mini-lab (20–30 min): - Intro-

Module 5 – Repositories, Namespaces, Values, Overrides (14:00–15:00)

Learning objectives:

- Work with multiple repos and namespaces.
- Override values using CLI and files.
- Understand precedence of values.

Review questions:

- What does `helm`

Final Project – Deploy a Small Web App (16:00–17:00)

Goal: Deploy a simple Node.js or Nginx web app using Helm with overrides and rollback.

Scenario: You need to deliver a minimal web front end. Use an existing chart, customize values, and demonstrate upgrade/rollback.

Steps: 1) Choose chart: `bitnami/nginx` (or `bitnami/node`). 2) Create project namespace: `kubectl create namespace final-web`. 3) Customize values file `final-values.yaml` (example):

```
image:
  repository: bitnami/nginx
  tag: 1.25
service:
  type: NodePort
  nodePorts:
    http: 30080
replicaCount: 2
resources:
  requests: {cpu: 50m, memory: 128Mi}
serverBlock: |
  server {
    listen 8080;
    location / {
      return 200 'Hello from Helm final project!';
    }
  }
```

4) Install: `helm install final-web bitnami/nginx -n final-web -f final-values.yaml --create-namespace`.
5) Verify: `kubectl get pods -n final-web`; `kubectl get svc -n final-web` and access via NodePort or `minikube service final-web -n final-web --url`.
6) Upgrade: change `replicaCount` to 3, run `helm upgrade final-web bitnami/nginx -n final-web -f final-values.yaml`.
7) Rollback: `helm rollback final-web 1 -n final-web` if upgrade causes issues.
8) Debugging: check `helm status final-web -n final-web`, `kubectl describe pod -n final-web`, `kubectl logs`.

Extra examples: - Swap the image to a simple Node.js sample (e.g., `bitnami/node`) and override `containerPorts` accordingly. - Add a small ConfigMap and mount it via `extraVolumeMounts` if using charts that support it (or use `serverBlock` for nginx as above). - Demonstrate `helm get all final-web -n final-web` to review everything installed.

Completion criteria: Service reachable, rollout status successful, history shows revisions, rollback tested.

Review questions: - How did values overrides change the deployment?
- What did the upgrade and rollback do to the release history?
- How would you expose this app in production (Ingress hint)?

Reference Commands & Cheatsheet

- Repo: `helm repo add <name> <url>`, `helm repo update`, `helm search repo <term>`
- Install/Upgrade: `helm install <release> <chart> [-f file.yaml] [--set k=v]`, `helm upgrade <release> <chart>`
- Status/History: `helm list -A`, `helm status <release>`, `helm history <release>`, `helm rollback <release> <rev>`
- Template/Lint: `helm template <chart>`, `helm lint <chart>`
- Uninstall: `helm uninstall <release>`
- Namespace: add `-n <ns> [--create-namespace]`
- Dry run: `helm install ... --dry-run --debug`