**Lab: Golden Path Demo using Cookiecutter Template**

**🎯 Goal:**

Enable developers to bootstrap new services in a **standardized and reproducible** manner using a **Golden Path** powered by **Cookiecutter**, and integrate CI/CD scaffolding, Docker, Helm, and observability—all from a single template.

**🧪 Summary**

| **Attribute** | **Details** |
| --- | --- |
| 🧑‍🏫 Type | Hands-on, instructor-guided (local or cloud-based terminal) |
| 🧰 Tools Used | Cookiecutter, GitHub CLI, Docker, Helm, GitHub Actions |
| 📁 Provided Inputs | Pre-configured Golden Path template repo |
| 🛠️ Output | Scaffolding of a microservice project with full CI/CD setup |

**✅ Step-by-Step: Create Your Golden Path Cookiecutter Template Repository**

**🧱 1. Directory Structure**

You’ll create a folder structure like this:

css

CopyEdit

cookiecutter-golden-path/

├── {{cookiecutter.project\_slug}}/

│ ├── app/

│ │ └── main.py

│ ├── Dockerfile

│ ├── helm/

│ │ ├── Chart.yaml

│ │ ├── templates/

│ │ │ └── deployment.yaml

│ ├── prometheus\_exporter.py

│ ├── .github/

│ │ └── workflows/

│ │ └── ci.yml

│ ├── README.md

│ ├── requirements.txt

│ └── tests/

│ └── test\_app.py

├── cookiecutter.json

└── LICENSE

**✨ 2. Create the Template**

**👉 Create base directory**

bash

CopyEdit

mkdir cookiecutter-golden-path && cd $\_

**👉 cookiecutter.json**

This defines the prompts users will see when generating a project:

json

CopyEdit

{

"project\_name": "My Microservice",

"project\_slug": "my\_microservice",

"author\_name": "Raman",

"description": "A sample microservice scaffolded using Cookiecutter.",

"port": "8080"

}

**🐍 3. Core Files and Templates**

**🔹 {{cookiecutter.project\_slug}}/app/main.py**

python

CopyEdit

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route("/")

def index():

return "Hello from {{ cookiecutter.project\_name }}!"

**🔹 {{cookiecutter.project\_slug}}/Dockerfile**

dockerfile

CopyEdit

FROM python:3.10-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

CMD ["python", "app/main.py"]

**🔹 {{cookiecutter.project\_slug}}/requirements.txt**

txt

CopyEdit

flask

prometheus\_client

**🔹 {{cookiecutter.project\_slug}}/prometheus\_exporter.py**

python

CopyEdit

from prometheus\_client import start\_http\_server, Counter

import time

REQUEST\_COUNT = Counter('app\_requests\_total', 'Total app HTTP requests')

if \_\_name\_\_ == "\_\_main\_\_":

start\_http\_server({{ cookiecutter.port }})

while True:

REQUEST\_COUNT.inc()

time.sleep(1)

**🔹 {{cookiecutter.project\_slug}}/.github/workflows/ci.yml**

yaml

CopyEdit

name: CI

on: [push, pull\_request]

jobs:

build:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v3

- name: Set up Python

uses: actions/setup-python@v4

with:

python-version: '3.10'

- name: Install dependencies

run: |

pip install -r {{ cookiecutter.project\_slug }}/requirements.txt

- name: Run Tests

run: |

pytest {{ cookiecutter.project\_slug }}/tests

**🔹 {{cookiecutter.project\_slug}}/helm/Chart.yaml**

yaml

CopyEdit

apiVersion: v2

name: {{ cookiecutter.project\_slug }}

description: A Helm chart for deploying {{ cookiecutter.project\_name }}

version: 0.1.0

**🔹 {{cookiecutter.project\_slug}}/helm/templates/deployment.yaml**

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: {{ cookiecutter.project\_slug }}

spec:

replicas: 1

selector:

matchLabels:

app: {{ cookiecutter.project\_slug }}

template:

metadata:

labels:

app: {{ cookiecutter.project\_slug }}

spec:

containers:

- name: {{ cookiecutter.project\_slug }}

image: your-docker-image

ports:

- containerPort: {{ cookiecutter.port }}

**{{cookiecutter.project\_slug}}/tests/test\_app.py**

def test\_example():

    assert 1 == 1

* **🔹 {{cookiecutter.project\_slug}}/README.md**

md

CopyEdit

# {{ cookiecutter.project\_name }}

This is a microservice scaffolded using the Golden Path Cookiecutter template.

## Run locally

```bash

python app/main.py

**Metrics endpoint**

bash

CopyEdit

curl http://localhost:{{ cookiecutter.port }}/metrics

yaml

CopyEdit

---

## 🧪 4. \*\*Local Test of the Template\*\*

After setting up your directory, you can test it:

```bash

# Outside the template directory

cookiecutter ./cookiecutter-golden-path

Follow the prompts and a new folder like orders-api will be created with your project scaffolded.

**☁️ 5. Push to GitHub**

Create a public repo:

bash

CopyEdit

cd ..

git init

gh repo create golden-path-cookiecutter-template --public --source=. --push

**Install in a virtual environment (Clean Isolation) :**

If you're working on multiple projects and want to isolate tools:

bash

CopyEdit

Apt update -y

sudo apt install python3-venv -y

python3 -m venv venv

source venv/bin/activate

pip install cookiecutter

cookiecutter --version

**🧩 6. Use in Training**

Once published, your learners can now run:

Clone the repository..

**✅ Step 1: Clone & Use the Cookiecutter Template (10 mins)**

(venv) root@ip-172-31-14-172:~# ls

cookiecutter-golden-path snap venv

(venv) root@ip-172-31-14-172:~# cookiecutter ./cookiecutter-golden-path/

📌 When prompted, enter:

* Service name: orders-api
* Programming language: python
* Service type: api
* Port: 8080
* Enable monitoring: yes
* Enable CI/CD: yes

🎯 This generates a new local folder: orders-api/

**✅ Step 2: Explore the Generated Golden Path (10 mins)**

From the generated folder, walk participants through:

cd orders-api/

tree -L 2

| **Folder/File** | **Description** |
| --- | --- |
| Dockerfile | Container definition |
| helm/chart/ | Kubernetes Helm chart |
| .github/workflows/ci.yml | GitHub Actions CI/CD pipeline |
| prometheus\_exporter.py | Dummy metric exporter |
| README.md | Pre-filled doc template |
| tests/ | Sample test scaffolding |

**✅ Step 3: Push to GitHub and Trigger CI (15 mins)**

bash

CopyEdit

# Create a new GitHub repo

gh repo create orders-api --public --source=. --push

✅ Navigate to **GitHub Actions tab** – CI pipeline runs automatically.

Expected steps:

* Code lint + test
* Build Docker image (simulated)
* Push to GHCR (or DockerHub – optional)
* Run security check (e.g., trivy, snyk – mocked)
* Check Prometheus exporter presence

**✅ Step 4: Add Observability Touchpoint (10 mins)**

**PROMOTHEUS\_EXPORTER.PY IS CREATING A STANDALONE WEBSERVER(http\_start\_server) FOR MOCKING/SIMULATING METRIC GENERATION IN THIS LAB , WE WILL INTEGRATE IT INSIDE OUR CODE IN NEXT LAB WITH THE FLASK app.**

If template includes prometheus\_exporter.py:

bash

CopyEdit

# Run service locally (optional)

python prometheus\_exporter.py

curl http://localhost:8080/metrics

✅ Validate:

* Metrics endpoint exists
* **Returns mock metric** (e.g., service\_up 1)

**✅ Step 5: Deploy to Kubernetes (Optional / Cloud Teams) (15 mins)**

For those with Helm/K8s cluster access

bash

CopyEdit

# Replace values.yaml if needed

helm install orders-api ./helm/chart/ --values values.yaml

kubectl get pods

kubectl get svc

🎯 Validate that the Helm chart works for a quick deploy.

**📊 Expected Outcomes**

| **Outcome** | **Evidence** |
| --- | --- |
| Service bootstrapped from template | orders-api/ folder + GitHub repo |
| CI/CD pipeline in place | GitHub Actions status check |
| Observability baked in | Prometheus /metrics endpoint |