SAMIR GENAI ACADEMY

PROFESSIONAL STUDY GUIDE SERIES

Phase 05 · Capstone & Deployment

24

Cloud Deployment & Scaling

AWS Bedrock, GCP Vertex, Azure AI — production deployment

Module 24 of 25 · **1 hour**

Instructor: **Samir Kumar** · Edition 2026

STUDY GUIDE

01 Overview

> AWS Bedrock, GCP Vertex, Azure AI — production deployment. This guide gives you the core theory, the tools that matter, a hands-on build, working code, and the pitfalls to avoid — everything you need to learn the topic to a professional standard and apply it in real projects.

1 hour	4 + 4	4	24/25
LESSON LENGTH	CONCEPTS + BUILD STEPS	FRAMEWORKS & TOOLS	PHASE 05

02 Learning Objectives

- Containerise an agent for reproducible deployment.
- Compare managed model platforms (Bedrock, Vertex, Azure AI).
- Scale with autoscaling, queues, and load balancing.
- Deploy a dockerised agent to the cloud.

03 Core Concepts

Containerise first

Package the agent and dependencies in a Docker image for reproducible builds across dev, staging, and prod. Containers are the unit of deployment for every major cloud and orchestrator (ECS, Cloud Run, AKS, Kubernetes).

Managed model platforms

AWS Bedrock, GCP Vertex AI, and Azure AI Foundry offer multiple foundation models behind one API with enterprise security, VPC isolation, and data-residency controls — valuable when compliance and a single cloud bill matter more than always having the newest model.

Scaling patterns

Put a queue in front of slow LLM work, autoscale stateless workers on queue depth or CPU, and load-balance across replicas. Cache and rate-limit to protect upstream model quotas. Design stateless services so scaling is trivial.

Ops & cost at scale

Add health checks, blue-green/canary deploys, secrets management, and per-tenant cost tracking. Watch provider quota limits — they, not your servers, are often the real ceiling.

04 Visual Diagram

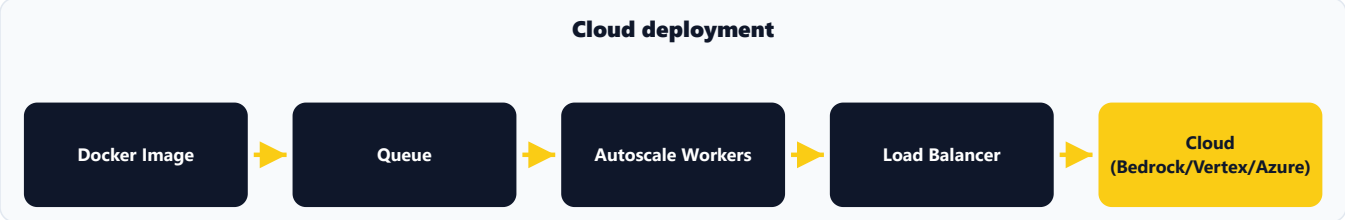

Figure 24 — Containerise, queue slow work, autoscale stateless workers, and front everything with a load balancer.

05 Key Frameworks & Tools

Docker | AWS Bedrock / ECS / Lambda | GCP Vertex / Cloud Run | Azure AI / Kubernetes

06 Hands-On Build

🛠 Dockerized Agent Deployment on AWS/GCP

1. Write a Dockerfile and build the agent image.
2. Add a queue for long-running tasks and stateless workers.
3. Deploy to Cloud Run / ECS with autoscaling.
4. Add health checks, secrets, and cost/quota monitoring.

07 Code Walkthrough

A production-ready agent Dockerfile — DOCKERFILE

```
FROM python:3.12-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
ENV PORT=8080
EXPOSE 8080
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8080"]
```

08 💉 Lab Assistant

Lab 24 · Dockerise & deploy

Prerequisites: Docker installed · A cloud account (Cloud Run / ECS)

1. **Write a Dockerfile and build the image**
   > Expected: A runnable container exposing port 8080
2. **Deploy to Cloud Run / ECS with autoscaling**
   > Expected: A public URL serving the agent
3. **Add health checks + secret manager + cost alert**
   > Expected: Production-ready ops

Troubleshooting

- Can't scale → make workers stateless.
- Quota errors → add a queue + backpressure.

09 Pitfalls & Key Takeaways

⚠ COMMON PITFALLS

- Stateful workers that can't scale horizontally.
- Hitting provider quota limits with no backpressure or queue.
- Baking secrets into images instead of a secret manager.

✓ KEY TAKEAWAYS

- Containerise for reproducible, portable deployment.
- Managed platforms trade cutting-edge models for compliance and ops ease.
- Queues + stateless autoscaling + caching scale AI services.

📺 **Companion video:** Watch the module 24 lesson in your Student Dashboard, then use this guide to revise, copy the code, and complete the build. Download both for offline study.

10 Further Resources

AWS Bedrock / GCP Vertex / Azure AI docs • Cloud Run & ECS guides • Docker best practices