Cloud Computing Final Year Project Ideas PDF

A good cloud computing final year project designs and deploys a working system on AWS, Azure or Google Cloud, using managed services, infrastructure as code, monitoring and a cost estimate. A serverless file processing app, a Kubernetes microservices deployment, a CI/CD pipeline and a cloud data lake are strong choices that fit free tiers with care.

Key takeaways

- Show the architecture diagram and explain each service choice.
- Use infrastructure as code (Terraform or CloudFormation) so the setup is repeatable.
- Add monitoring, logging and a monthly cost estimate.
- Set billing alerts on day one and delete resources after the demo.
- The PDF gives plans and steps, not finished source code.

Who this PDF is for

This PDF is for final-year B.Tech, BCA and MCA students who want a cloud project that goes beyond hosting a website. Each plan lists services (examples use AWS names; Azure and GCP have equivalents), modules a team of 3 or 4 can split (application, infrastructure, pipeline, monitoring), and viva points. Most can run within free-tier limits if you are careful with resource sizes and shut things down.

How to choose your project

- **One cloud provider**, the one your course or certification uses.
- **Billing alerts first.** Set a small budget alert before creating anything.
- **Prefer serverless and managed services** for lower cost and less admin.
- **Plan the demo** with a teardown and redeploy using your IaC scripts.

Standard level projects

1. Serverless Image Processing Pipeline

- **Problem:** Resize and tag images automatically on upload.
- **Cloud services:** AWS S3, Lambda, DynamoDB, API Gateway (or Azure Blob and Functions).
- **Tech stack:** Python, AWS SAM or Terraform, a React upload page.
- **Main modules:** Upload API, S3 trigger, Lambda processing, metadata store, gallery.
- **Steps:** Create buckets, write the Lambda, trigger on upload, save metadata, show a gallery.

- **Show in the viva:** Why serverless costs nothing when idle.
- **Extensions:** Add image labels with a vision API.

2. Three-tier Web App with Auto Scaling

- **Problem:** Host a web app that scales with traffic.
- **Cloud services:** AWS VPC, EC2 Auto Scaling, Application Load Balancer, RDS.
- **Tech stack:** Terraform, Node.js or Django app, k6 for load tests.
- **Main modules:** VPC with public and private subnets, scaling group, load balancer, database, load test.
- **Steps:** Write Terraform for the network and servers, deploy, run a load test, watch scaling.
- **Show in the viva:** Why the database sits in a private subnet.
- **Extensions:** Add a CDN with CloudFront.

3. CI/CD Pipeline for a Containerised App

- **Problem:** Build, test and deploy on every push.
- **Cloud services:** GitHub Actions, Docker, AWS ECR and ECS or a Kubernetes cluster.
- **Tech stack:** Docker, GitHub Actions, Trivy for image scanning.
- **Main modules:** Build and test stage, image scan, push to registry, deploy, rollback.
- **Steps:** Write the Dockerfile, set up the workflow, scan images, deploy, practise a rollback.
- **Show in the viva:** The pipeline stages and what fails a build.
- **Extensions:** Add blue-green deployment.

4. Cloud-based File Sharing with Expiring Links

- **Problem:** Share files securely with links that expire.
- **Cloud services:** AWS S3 presigned URLs, Cognito, Lambda.
- **Tech stack:** Python or Node.js, React.
- **Main modules:** Login, upload, presigned link generation, expiry, access log.
- **Steps:** Set up user login, upload to a private bucket, generate presigned links with expiry.
- **Show in the viva:** How presigned URLs avoid making buckets public.
- **Extensions:** Add virus scanning on upload.

Advanced level projects

5. Microservices on Kubernetes

- **Problem:** Deploy a multi-service app with service discovery and scaling.
- **Cloud services:** Amazon EKS, Azure AKS or a local kind or minikube cluster.
- **Tech stack:** Docker, Kubernetes, Helm, Prometheus, Grafana.
- **Main modules:** Three services, Helm charts, ingress, autoscaling, monitoring dashboards.
- **Steps:** Containerise services, write Helm charts, deploy, set up autoscaling and dashboards.
- **Show in the viva:** Pods, services and deployments in simple words.
- **Extensions:** Add a service mesh.

6. Cloud Data Lake and Analytics

- **Problem:** Store raw data cheaply and query it with SQL.
- **Cloud services:** AWS S3, Glue, Athena, QuickSight; public data such as NYC TLC trip records.
- **Tech stack:** Python, SQL, Terraform.
- **Main modules:** Raw and curated zones, ETL job, data catalogue, SQL queries, dashboard.
- **Steps:** Load raw files, convert to Parquet with Glue, query with Athena, build a dashboard.
- **Show in the viva:** Why columnar formats cut query cost.
- **Extensions:** Add scheduled ingestion.

7. Multi-region Disaster Recovery Setup

- **Problem:** Keep an app running if one region fails.
- **Cloud services:** AWS Route 53 health checks, S3 cross-region replication, RDS snapshots.
- **Tech stack:** Terraform, a simple web app.
- **Main modules:** Primary and standby regions, replication, health-check failover, recovery runbook.
- **Steps:** Deploy in two regions, replicate data, simulate failure, measure recovery time.
- **Show in the viva:** RTO and RPO with your measured numbers.
- **Extensions:** Automate failback.

8. Cost Monitoring and Optimisation Dashboard

- **Problem:** Track cloud spend and suggest savings.
- **Cloud services:** AWS Cost Explorer API, CloudWatch, Lambda.
- **Tech stack:** Python, boto3, a dashboard tool.
- **Main modules:** Cost data collection, idle resource finder, rightsizing tips, alerts.
- **Steps:** Pull daily cost data, flag idle volumes and stopped instances, send weekly reports.
- **Show in the viva:** How tagging helps cost reports.
- **Extensions:** Add auto-stop schedules for dev servers.

Research-oriented projects

9. Serverless vs Container Performance Study

- **Problem:** Compare latency and cost of the same API on Lambda and containers.
- **Cloud services:** AWS Lambda, ECS Fargate or Kubernetes.
- **Tech stack:** Node.js or Python, k6, CloudWatch.
- **Main modules:** Same API on both, load tests, cold start measurement, cost model.
- **Steps:** Deploy both, run identical load tests, measure cold starts and cost per million requests.
- **Show in the viva:** Where each option wins.
- **Extensions:** Add a third option on a VM.

10. Policy as Code for Cloud Security

- **Problem:** Block insecure infrastructure before deployment.
- **Cloud services:** Terraform code for test resources.
- **Tech stack:** Open Policy Agent or Checkov, GitHub Actions.

- **Main modules:** Policies (no public buckets, encryption on), pipeline check, report.
- **Steps:** Write policies, run them on Terraform plans in CI, show blocked changes.
- **Show in the viva:** Shift-left security in simple words.
- **Extensions:** Add drift detection.

11. Edge and Cloud IoT Data Platform

- **Problem:** Collect sensor data at the edge and analyse it in the cloud.
- **Cloud services:** AWS IoT Core, Kinesis or Timestream; ESP32 or a simulated device.
- **Tech stack:** MQTT, Python, Grafana.
- **Main modules:** Device simulator, IoT rules, stream storage, dashboards, alerts.
- **Steps:** Simulate devices, route messages with IoT rules, store and chart data.
- **Show in the viva:** Edge versus cloud processing trade-offs.
- **Extensions:** Add anomaly detection.

12. Autoscaling Policy Comparison

- **Problem:** Test which scaling rules handle traffic spikes best.
- **Cloud services:** Kubernetes HPA on a local or cloud cluster.
- **Tech stack:** Kubernetes, k6, Prometheus.
- **Main modules:** Test app, three scaling policies, spike tests, metrics.
- **Steps:** Run spike tests under CPU, request-rate and scheduled scaling, compare latency and cost.
- **Show in the viva:** Which policy handled spikes best, with charts.
- **Extensions:** Try KEDA for event-driven scaling.

Report structure

A final-year report is marked on clarity as much as on code. A structure most universities accept:

1. **Abstract** (150 to 250 words): problem, method, main result.
2. **Introduction:** background, problem statement, objectives and scope.
3. **Literature review:** 8 to 15 papers or existing systems, and the gap your project fills.
4. **Requirements and design:** functional and non-functional requirements, architecture diagram, the cloud architecture diagram, IaC structure, security groups and IAM roles.
5. **Implementation:** modules, tools, key algorithms and screenshots.
6. **Testing and results:** load or recovery test results, monitoring screenshots, and a monthly cost estimate from the provider's pricing calculator.
7. **Conclusion and future scope.**
8. **References** in the format your department asks for (IEEE style is common).

Follow your own university's template if it gives one; it overrides this list.

Common mistakes to avoid

- No billing alert, leading to a surprise bill.
- Using the root account and wide IAM permissions.

- Creating everything by clicking in the console, so it cannot be rebuilt.
- Forgetting to delete resources after the demo.

Frequently asked questions

Which cloud computing project is best for a final year?

A serverless pipeline, a Kubernetes microservices deployment, a CI/CD pipeline or a data lake. Each shows architecture design, automation and monitoring.

AWS, Azure or Google Cloud for a final year project?

Any of them. Pick the one your course or certification uses. AWS has the most student tutorials.

Can I do a cloud project for free?

Often yes, within free-tier limits, if you use small resources and delete them after use. Set a billing alert first.

What is infrastructure as code?

Writing your cloud setup in files (Terraform or CloudFormation) so it can be created, changed and deleted with commands instead of clicks.

Does this PDF include full source code?

No. It gives the problem, data, stack, modules and steps for each project. You write the code yourself, which is what examiners test in the viva.

Is this PDF free?

Yes. Fill in the short form on this page and the PDF downloads right away.

Sources and further reading

- AWS Free Tier
- AWS Well-Architected Framework
- Kubernetes documentation
- Terraform documentation

About this PDF

Prepared by the Uncodemy training team as practice project ideas. Datasets and tools named are public; check each dataset's licence before use. We update it every January. Corrections: info@uncodemy.com.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550