AI-Driven Cisco Modeling Labs on Google Cloud Platform

Nested Virtualization Lab Factory for CCNA Practice, AI Automation, and Infrastructure as Code

Project Artifact for Social / Portfolio Use

Planned with ChatGPT → Built with Claude Desktop → Deployed with Google Anti Gravity on Google Cloud

EXECUTIVE SUMMARY

This project delivers an AI-driven Cisco Modeling Labs (CML) platform on Google Cloud Platform that combines intelligent planning, automated infrastructure, and nested virtualization to create limitless, on-demand CCNA practice environments. ChatGPT is used for solution design and lab scenario planning, Claude Desktop handles coding and orchestration, and Google Anti Gravity accelerates deployment and operations on Google Cloud. AI-assisted automation generates Packet Tracer-style labs with hidden break/fix faults aligned to CCNA exam objectives—enabling scalable, repeatable, and cloud-native networking lab experiences.

1. AI PLANNING
- Requirements
- Lab scenarios
- Architecture design

2. AI BUILD & ORCHESTRATE
- Code generation
- IaC & automation
- Configuration as Code

3. DEPLOY TO GCP
- Google Anti Gravity
- Terraform provisioning
- Nested virtualization

4. CML LAB FACTORY
- Cisco Modeling Labs
- Hidden break/fix faults
- On-demand environments

5. PRACTICE & IMPROVE
- Troubleshoot & validate
- Map to CCNA objectives
- Track progress

PROJECT GOALS

- Automate Cisco lab creation in the cloud
- Practice CCNA troubleshooting with hidden faults
- Build repeatable labs using Terraform, YAML, and Python APIs
- Strengthen skills in virtualization, containerization, and IaC
- Create a reusable AI automation learning platform

CORE STACK

ChatGPT · Claude Desktop · Google Cloud Platform · Google Anti Gravity · Cisco Modeling Labs · Nested Virtualization · Terraform · YAML · Python API · Containers

QUICK FACTS

Platform:	Google Cloud Platform
Region / Zone:	us-central1 / us-central1-a
VM Profile:	n2-standard-4
CPU Platform:	Intel Cascade Lake
Storage:	100 GB SSD
Access:	HTTPS, SSH, REST API
Primary Outcome:	On-demand CML lab generation

System Architecture & Technical Design

How AI planning, cloud infrastructure, and nested virtualization work together

1. PROMPT *Plan*

2. CODE *Generate*

3. DEPLOY *Provision*

4. LAB *Generate*

5. LEARN *Outcomes*

1. USER / AI LAYER

- **ChatGPT** — AI Planning: Requirements & Lab Scenarios
- **Claude Desktop** — Coding & Orchestration: IaC, Automation, Configuration
- **Network Engineer / Student** — Define Objectives, Review, Iterate

2. AUTOMATION LAYER

- **Terraform** — Infrastructure Provisioning (IaC)
- **YAML** — Configuration as Code
- **Python API** — REST API Automation
- **Google Anti Gravity** — Workflow Orchestration

3. GOOGLE CLOUD LAYER

GCP Project: (project ID redacted)
Region: us-central1

- **VPC** — cml-network, 10.10.0.0/16
- **Firewall** — cml-allow-ui-ssh, Allow: 22, 443
- **Compute Engine VM** — cml-controller (External IP)

4. VM / HYPERVISOR LAYER

Compute Engine VM: cml-controller

- n2-standard-4, 4 vCPU, 16 GB RAM
- Intel Cascade Lake CPU
- Nested Virtualization Enabled
- Ubuntu / CML Appliance OS

CML Controller Service

HTTPS :443 | SSH :22 | REST API

5. LAB FACTORY LAYER

KVM / QEMU Hypervisor (Nested)

- Routers (vIOSv)
- Switches (vIOSvL2)
- Endpoints (PCs)
- VLANs (Multi-VLAN)
- STP (Topology)
- Hidden Break/Fix Fault Injected

Packet Tracer-Style Troubleshooting Labs (On-Demand)

6. LEARNING OUTPUT LAYER

- **Student Troubleshooting Practice** — Hands-on labs with hidden faults
- **Instructor Notes & Guides** — Solutions, validations, and explanations
- **Repeatable CCNA-Aligned Exercises** — Scalable, consistent, exam-focused

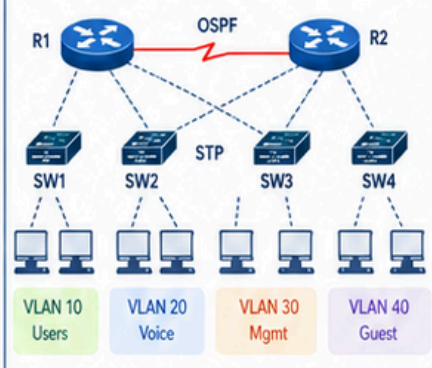
TECHNICAL HIGHLIGHTS

- Nested virtualization on Google Compute Engine
- On-demand CML topology creation
- REST API and Python orchestration
- Terraform-based infrastructure provisioning
- Repeatable YAML / configuration-driven builds
- Hidden-fault troubleshooting scenarios

DESIGN CONSTRAINTS & DECISIONS

- Public HTTPS / SSH used for initial development
- VM can be stopped to save compute cost
- SSD chosen for router boot performance
- Architecture built for reuse and scale

REFERENCE TOPOLOGY EXAMPLE



Hidden Fault Example: SW2 Gi0/1 Port Flap / STP Root Inconsistency

Deployment Workflow & Project Plan

From image import to live lab orchestration

Project Artifact for Social / Portfolio Use

1. DEPLOYMENT WORKFLOW

1. Prepare CML appliance image
2. Import image into Google Cloud
3. Provision VPC, firewall, and VM with Terraform
4. Enable nested virtualization on n2-standard-4
5. Start CML Controller and expose HTTPS / SSH / API
6. Run Python orchestration with virl2_client
7. Create lab, nodes, links, startup configs
8. Launch troubleshooting topology

2. PROJECT STATUS & PLAN

PHASE / DELIVERABLE	STATUS
GCP Project Setup	COMPLETED
CML Appliance Image Import	SUCCESS
Network & Firewall IaC	PROVISIONED
Compute Controller VM (n2-standard-4)	PROVISIONED *(Currently Stopped)*
Python API Orchestration	RESOLVED
Next Phase	Hidden-Fault Lab Templates & Expansion

3. DELIVERY MILESTONES

1. Infrastructure as Code foundation
2. Secure cloud network setup
3. Nested virtualization activation
4. Automated topology generation
5. Reusable training-lab pipeline

4. OPERATIONAL RUNBOOK

1. Start controller VM
2. Retrieve public endpoint
3. Stop controller when inactive to save cost

5. KEY ARTIFACTS PRODUCED

- Terraform code
- YAML definitions
- Python automation
- System design document
- Stakeholder presentation

QUICK NOTES

- Built on Google Cloud Platform with Infrastructure as Code
- Secure by design with firewall rules and least-privilege access
- Automated, repeatable, and cost-optimized operations

Learning Outcomes, Skills Growth & CCNA Alignment

What this project teaches and why it matters

Project Artifact for Social / Portfolio Use

1 SKILLS MASTERED THROUGH THE PROJECT

VIRTUALIZATION
nested KVM, cloud-hosted hypervisors, lab lifecycle management

CCNA NETWORKING
VLANs, STP, trunks, routing, troubleshooting workflows

CONTAINERIZATION
understanding platform packaging and automation workflows

TERRAFORM
repeatable infrastructure provisioning

YAML
declarative configuration design

AI AUTOMATION
prompt-driven planning, coding, orchestration, and validation

INFRASTRUCTURE AS CODE
versioned, repeatable, documentable systems

2 CCNA OBJECTIVE MAPPING

CCNA DOMAIN	HOW THIS PROJECT ADDRESSES IT
Network Fundamentals	topology planning, addressing, virtualization context
Network Access	VLANs, trunks, STP, switch behavior
IP Connectivity	router-on-a-stick, reachability, routing validation
IP Services	management access, automation APIs, service readiness
Security Fundamentals	firewall rules, controlled access, hardening considerations
Automation & Programmability	Terraform, Python APIs, YAML, AI-assisted workflows

3 HIDDEN BREAK/FIX LAB ENGINE

Endless practice. Every lab is unique.

Prompt → Topology Template → Automatic Configuration → Hidden Fault Injection → Student Troubleshooting → Validation and Review

Example Hidden Faults

- Native VLAN mismatch
- Trunk allowed VLAN omission
- Wrong STP root placement
- Shutdown access port
- Incorrect default gateway
- Misconfigured router subinterface

WHY THIS PROJECT IS PORTFOLIO-READY

- Demonstrates real cloud deployment
- Shows network automation and IaC
- Connects AI tooling to infrastructure delivery
- Produces repeatable training value

Use Cases, Value, and Operational Impact

How this platform delivers practical learning and cloud efficiency

Project Artifact for Social / Portfolio Use

MAJOR USE CASES

1 CCNA Troubleshooting Lab Factory

Generate endless Packet Tracer-style labs with hidden faults for self-paced troubleshooting mapped to exam objectives.

- Hidden break/fix faults
- Mapped to CCNA objectives
- Auto-reset and re-seed scenarios
- Instant, repeatable practice

2 Instructor / Mentor Challenge Builder

Rapidly create guided labs, answer keys, and scenario-based exercises for demos, coaching, or study groups.

- Guided labs with solutions
- Scenario variations & branching
- Exportable instructions & keys
- Supports cohorts & workshops

3 Cloud-Based Network Sandbox

Safely test VLAN, STP, routing, and change scenarios before touching production networks.

- Isolated, on-demand environments
- Test change windows safely
- Rollback with one click
- Zero impact to production

COST PER DAY: ACTIVE VS STOPPED

30 days ending Jul 2, 3:00 AM

30-Day Project Cost

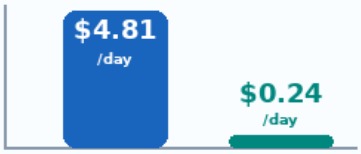
$59.05

Cost Change

-61.15% ↓

Latest GCP Cost Explorer view

Daily Cost by VM State



Save about $4.57 per idle day

While CML is UP

$4.81/day

Live labs + compute burn

While not in use

$0.24/day

VM stopped; disk kept

30-Day Cost Drivers

Compute Engine	$51.86 (97.8%)
Persistent Disk	$4.00 (6.8%)
Other Services	$3.19

Stop controller after labs to cut compute billing

BUSINESS / LEARNING VALUE

- Lower cost than permanent on-prem lab hardware
- Faster lab provisioning and reset cycles
- Repeatable, versioned, cloud-native practice environment
- Strong artifact for networking, cloud, and automation portfolios

NEXT STEPS

1. Expand topology templates
2. Add automated validation and grading
3. Integrate GitHub-based version control
4. Harden access with IAP and tighter firewall scopes

> ***This project turns cloud infrastructure into a reusable CCNA lab factory—bridging AI, automation, and hands-on network troubleshooting.***