Hann.

Web portfolio



Public Cloud Infrastructure & Security Engineer

Kim JunHyun

I turn infrastructure and security that people had to remember into systems that hold on their own.

2020~	4	3,500+	20+	15
start of cloud career	clouds operated	servers operated at peak	service projects	certifications earned

kimjhtlzk@naver.com +82 10-9092-0615 github.com/kimjhtlzk linkedin.com/in/ihanni hann-blume.tistory.com

Summary

From live games to financial services, running infrastructure across 4 clouds (AWS, GCP, Azure, and Tencent), at up to 3,500 servers and 650 databases. Moving to Terragrunt cut deployment time by 40%. CSPM (cloud security posture checks) remediation rose from 60% to 85% or higher.

Core strengths

IaC automation	Define infrastructure as code with Terraform and Terragrunt
Multi-cloud operations	Design and operate 4 cloud providers together
Cloud security	Detection and response, least privilege, and audit response
Operations automation	Remove repetitive work with Python and PowerShell
AI agents	Build multi-agent systems with defined roles and permissions

Experience

Woori FIS

Feb 2026 – Oct 2026

Network Security Team, Information Security Department
Assistant Manager · Cloud Security Engineer

Designed, built, and operated cloud security architecture for affiliates including Woori Bank, Woori Card, and Woori Financial Group, across a hybrid environment of public cloud, private cloud, and on-premises.

- Brought AWS Network Firewall, WAF, and Shield security infrastructure under code for the first time, in an environment with no Terraform
- Built a Python pipeline that automatically collects and analyzes network communication settings across multiple AWS accounts
- Integrated GuardDuty, Security Hub, WAF, Network Firewall, and Shield with a Splunk-based SIEM/SOAR
- Defined IAM permissions in Terraform to block excessive grants by design, following Zero Trust principles and the network separation (a Korean financial regulation) requirements in Korea's Electronic Financial Supervisory Regulations
- Operated hybrid security solutions together: FortiGate, NxPortrait, and WIPS
- Helped set security policy as an early member of public cloud adoption in the financial sector

Com2us Platform

Feb 2023 – Feb 2026

Infrastructure Technology Team C, Infrastructure Technology Office, Cloud Division
Assistant Manager · Cloud Engineer

Main engineer in charge of design, build, operations, and incident response for multi-cloud game service infrastructure across 4 cloud service providers (CSPs): AWS, GCP, Azure, and Tencent.

- Cut deployment time by 40% by moving from Terraform to Terragrunt
- Migrated from GCP to Tencent Cloud with zero downtime (80 servers and 20 databases, completed within one week)
- Raised CSPM (cloud security posture check) remediation from 60% to 85% or higher
- Cut DragonSky infrastructure costs by up to 62%
- Analyzed every cross-cloud firewall policy across Tencent and GCP, at 600 servers and 70 databases, then redesigned them
- Automated audits in Python that collect assets and audit logs from 4 CSPs into a common schema. Permission grants, Access Key issuance, and changes to key servers and databases trigger immediate Slack and Telegram alerts
- Handled audits across 4 CSPs with no security team working directly in the cloud: moved recurring findings into Terraform module defaults and mandatory validate and plan gates in CI/CD
- Ran unified alerting with Zabbix, Grafana, Slack, and Telegram, and DDoS defense with GCP Cloud Armor

C&Thoth

Aug 2020 – Feb 2023

Cloud Service Team
Associate to Assistant Manager · Cloud Engineer (Azure MSP)

Main engineer dedicated to game company clients at an Azure managed service provider (MSP), covering design, build, operations, security audits, and deployment automation.

- Dedicated operations for MIR4 Global game infrastructure: 3,500 servers and 650 databases worldwide
- Automated DDoS detection and blocking with Azure Log Analytics and PowerShell
- Improved deployment speed by 80% with PowerShell deployment automation (MIR4 Korea)
- High-availability architecture on Dedicated Host, and real-time incident response
- Built real-time dashboards with Telegraf, InfluxDB, and Grafana (TIG stack)
- Handled client cloud security audits based on CSPM and ISMS (Korea's information security management certification)

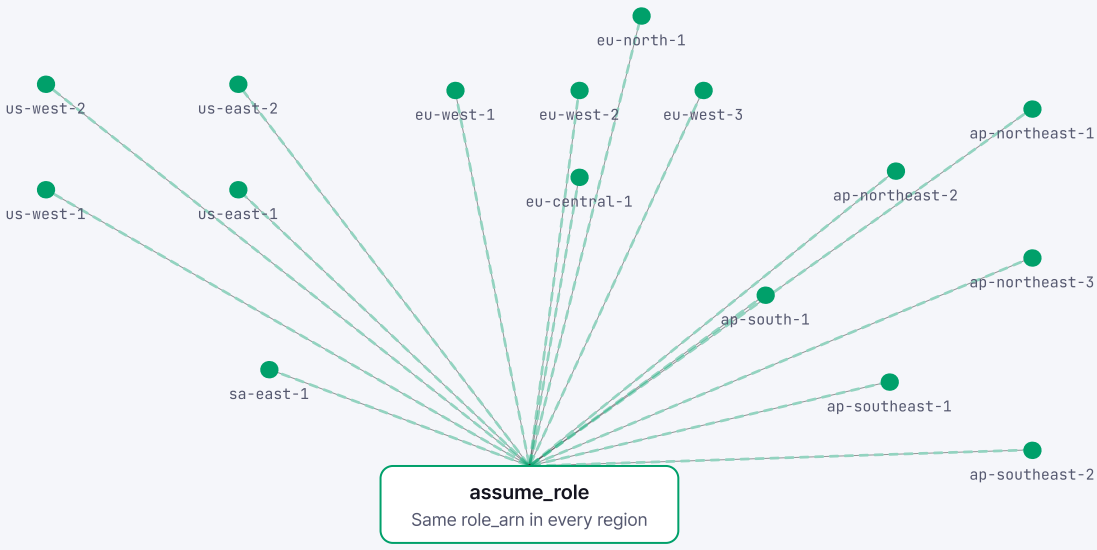
AWS multi-region Terraform

A Terraform module set that builds the same infrastructure in 16 AWS regions from one locals map per region.

16
Regional provider aliases

24
Terraform modules

3
Validated examples



16 regions · 24 modules · one locals map per region

Problem

The same resources had to exist in many regions. More regions meant more copied code and a higher risk that regions drift apart.

Approach

A locals map per region declares the wanted state, and module plus for_each creates it all. Each region has a provider alias that assumes the same role. Terraform cannot choose a provider dynamically inside for_each, so the module block repeats per region.

Result

A change is one map entry, and stable keys mean reordering never recreates anything. The 24 modules carry defaults for encryption, IMDSv2, and private S3, and three examples cover networks, EKS, and alerting.

Design and implementation · Personal project · released Nov 2026 · Terraform, AWS, EKS, CloudWatch, AWS Chatbot

github.com/kimjhtlzk/pf-01-aws-terraform-multi-region

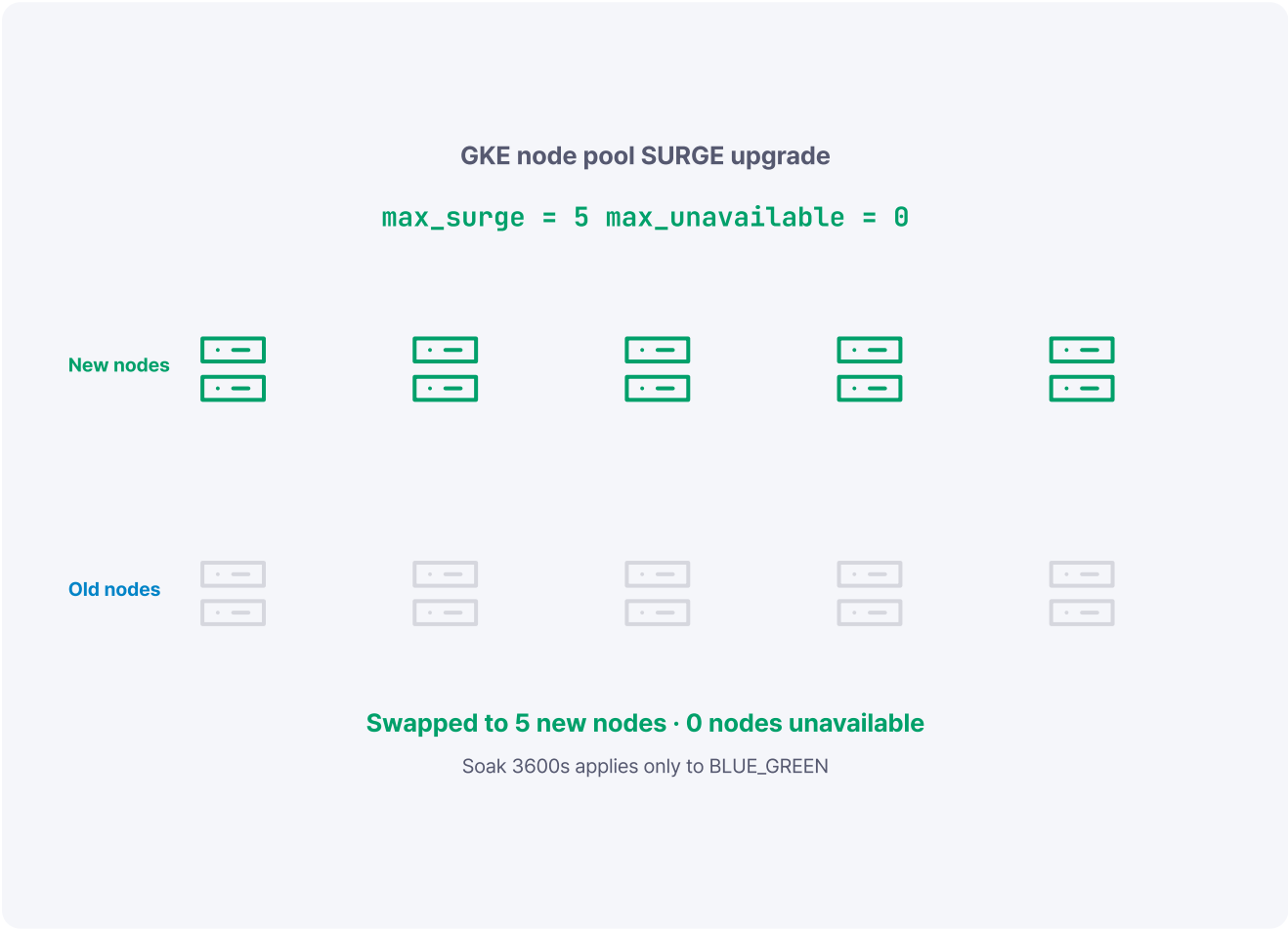
GCP Terraform GKE platform

A module set that keeps capacity during GKE node pool upgrades with SURGE, and opens only what a VPN needs through a firewall baseline, all as code.

22
Terraform modules

5
Extra nodes first (max_surge)

0
Nodes down at once (max_unavailable)



Problem

A GKE node upgrade must not reduce serving capacity, and management ports should open only from a VPN.

Approach

The node pool upgrades with SURGE: five new nodes come up first and unavailable stays at zero. The c2sfirewall module opens management ports from the VPN at priority 1000 and denies other tcp and udp outbound at priority 65534.

Result

Old nodes leave only after replacements are ready. Host errors flow from the hostError audit log to a log-based metric, an alert policy, and a Slack channel.

Design and implementation · Personal project · released Nov 2026 · Terraform, GCP, GKE, Cloud Logging, Cloud Monitoring

github.com/kimjhtlzk/pf-02-gcp-terraform-gke-platform

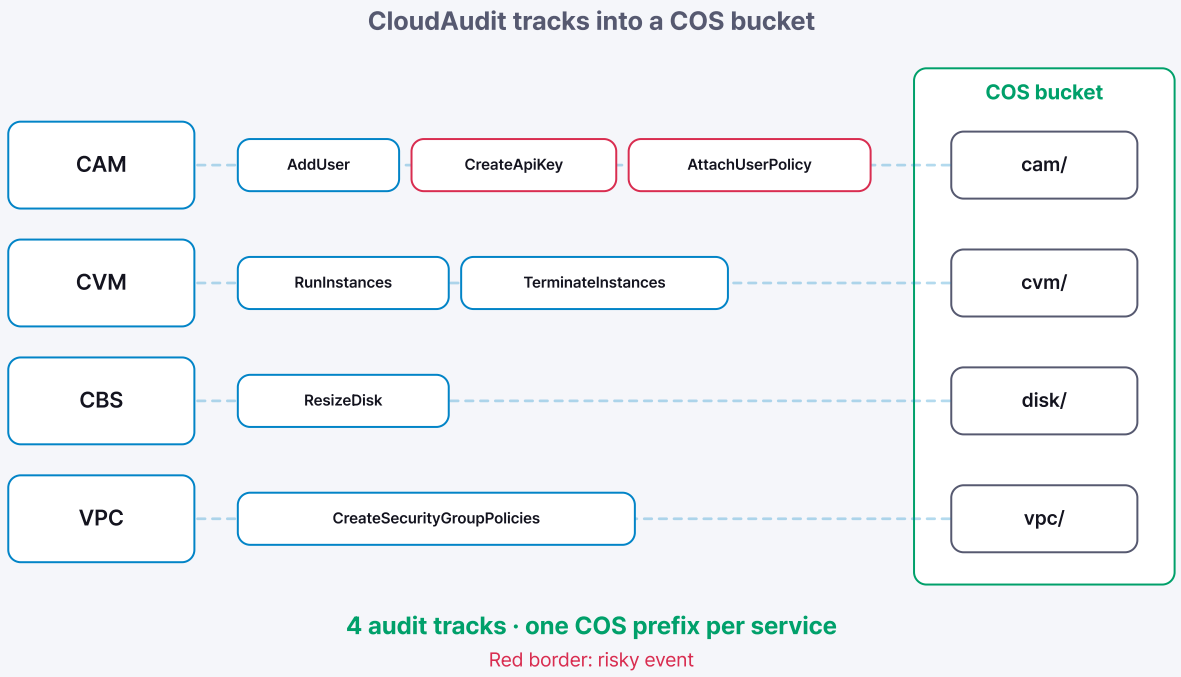
Tencent Cloud audit-ready Terraform

Tencent Cloud modules that define CloudAudit tracks per service, so risky changes land in a COS bucket.

18
Terraform modules

4
Audit tracks by service

19
Audit events tracked in code



Problem

Changes on Tencent Cloud needed a per-service record of who did what. The provider also ships on an irregular cycle, so a new version could change a plan without any code change.

Approach

CloudAudit tracks are defined per service in code for CAM, CVM, CBS, and VPC, and each stores to its own COS prefix. Every module pins the provider to = 1.81.163.

Result

Permission, server, disk, and security group changes reach COS through the tracks defined in code, including risky events such as CreateApiKey and AttachUserPolicy.

Design and implementation · Personal project · released Nov 2026 · Terraform, Tencent Cloud, CloudAudit, COS, CAM, CLB

github.com/kimjhtlzk/pf-03-tencent-terraform-audit-ready

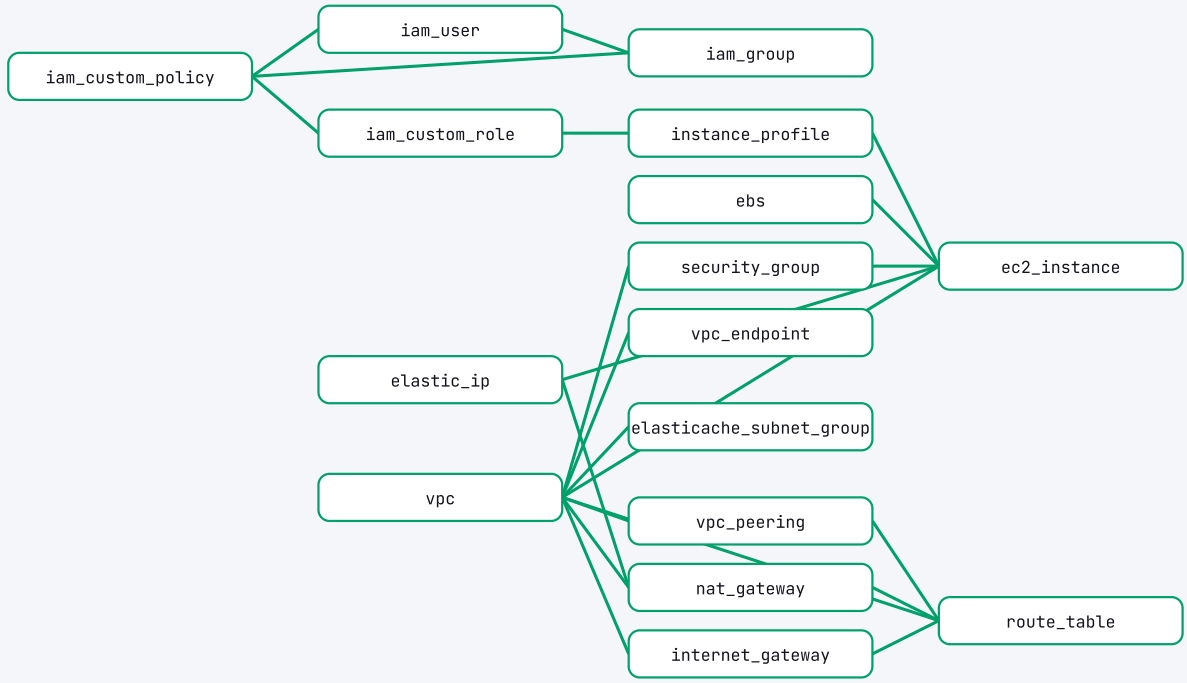
AWS Terragrunt DRY IaC

Sixteen Terraform modules behind one root config, with Terragrunt running them in order from their dependency graph.

16
Terraform modules and units

1
Root config replacing 16 copies

-40%
Deployment time (production result, not measured in this repository)



Problem

Each Terraform root repeated its backend and provider blocks, and the order between roots lived in people's heads.

Approach

Modules stay plain Terraform and Terragrunt wraps them. The backend, provider, and common inputs are generated once from live/root.hcl. _envcommon holds module sources and dependencies, and each unit states only its inputs. dependency blocks build the graph, and mock_outputs let the whole stack validate without a cloud account.

Result

This repository groups 16 modules and 16 units into four dependency levels and validates them without cloud credentials. The 40% shorter deployment time is a production result from moving a live estate to this pattern, and was not measured in this repository.

Design and implementation · Personal project · released Nov 2026 · Terragrunt, Terraform, AWS

github.com/kimjhtlzk/pf-04-aws-terragrunt-dry-iac

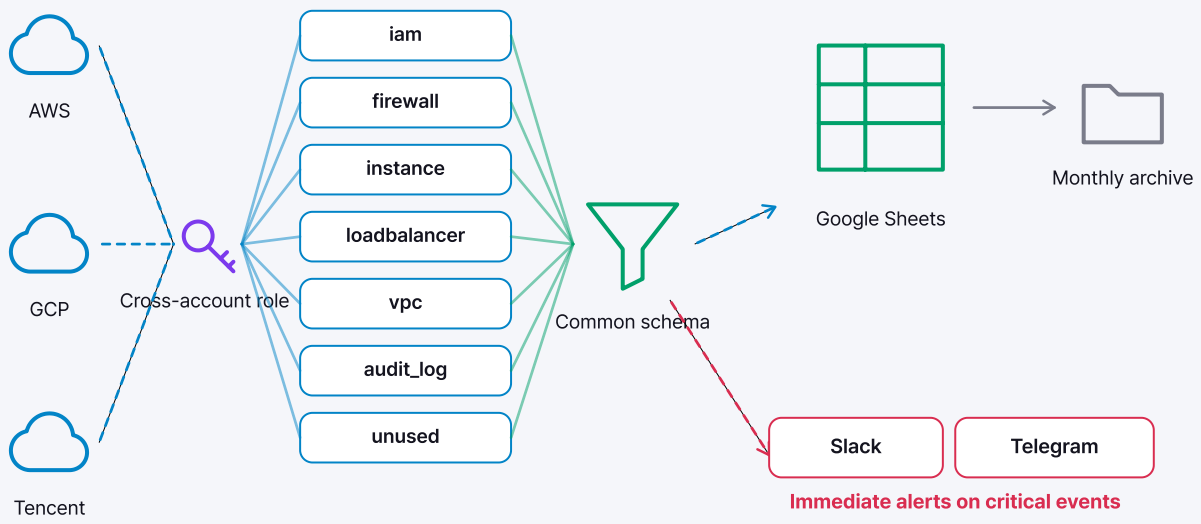
Multi-cloud asset and audit automation

A Python tool that collects assets, permissions, and audit logs from AWS, GCP, and Tencent Cloud into one schema, and alerts on risky events.

3
Clouds covered by the public code

7
Collectors by resource type

49
Audit actions tracked (16 alert immediately)



Problem

Each cloud returns the same concept in a different shape. Comparing permissions or firewalls meant reading three formats by hand.

Approach

Each account is reached through a cross-account role such as STS AssumeRole. Seven collectors, one per resource type, feed a single data class per resource. Output goes to Google Sheets and CSV, and 16 critical events alert Slack and Telegram immediately based on critical_events.yaml.

Result

The production tool behind this code checked four clouds including Azure every day before dawn, removed several days of manual month-end review, and together with Terraform defaults and CI gates helped raise the remediation rate from 60% to over 85%. Those figures are production results and were not measured in this repository.

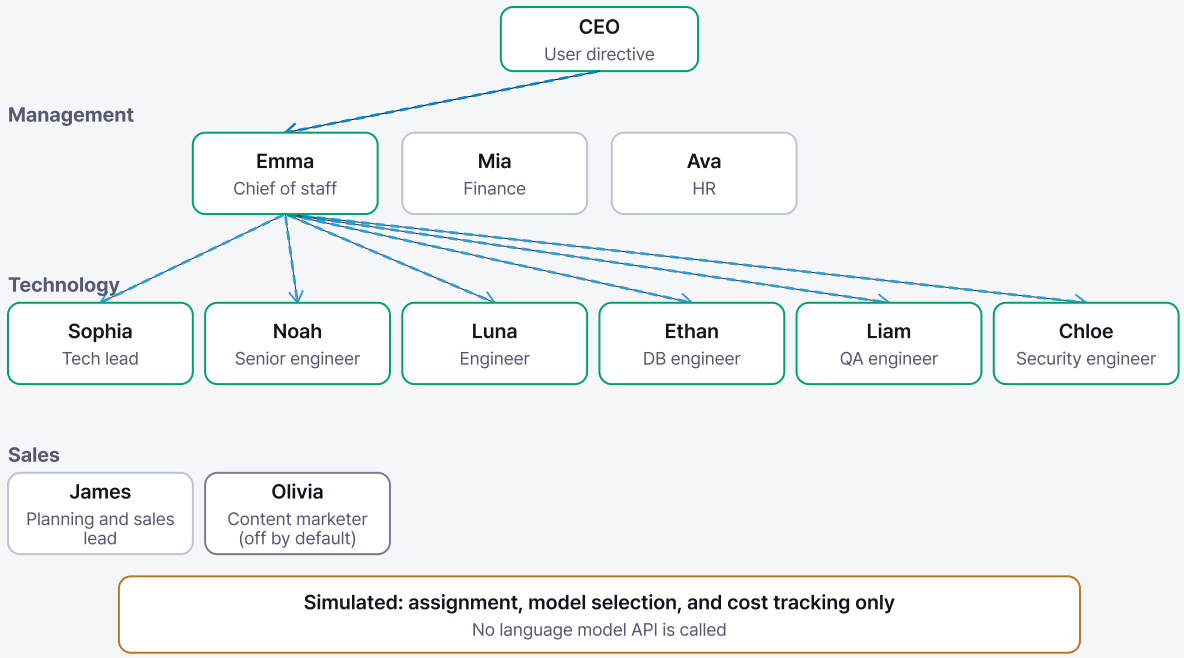
Design and implementation · Personal project · released Nov 2026 · Python, boto3, Google Cloud SDK, Tencent Cloud SDK, Google Sheets, Slack, Telegram

github.com/kimjhtlzk/pf-05-multicloud-asset-audit-automation

AetherCrew: a virtual company of 11 AI employees

A hand-built orchestrator that assigns work, picks models, and tracks cost for 11 AI employees in a virtual company. Agent work is simulated and no language model API is called.

11	3	8
AI employees	Divisions	Dashboard pages



Problem

When AI employees split work by role, code must control which model each one uses, how much has been spent, and what can be written to outside systems.

Approach

The chief of staff takes a CEO directive, writes a plan, and splits it into tasks by role. Each employee has a priority list of Anthropic, OpenAI, Google, and DeepSeek models and moves to the next on failure. Estimated costs add up, with warnings at 80% and 95% and work stopped at 100%. External writes are off by default, and file access stays inside the project root.

Result

Eight FastAPI dashboard pages show the org chart, cost, logs, and a pixel-art CCTV office. It runs with no API keys and calls no language model, so it produces no real work output. The Slack, Jira, and GitHub clients are real code and run only when INTEGRATIONS_LIVE_WRITE is on.

Design and implementation · Personal project · released Nov 2026 · Python, FastAPI, WebSocket, pytest, Slack, Jira, GitHub

github.com/kimjhtlzk/pf-06-aethercrew-ai-company

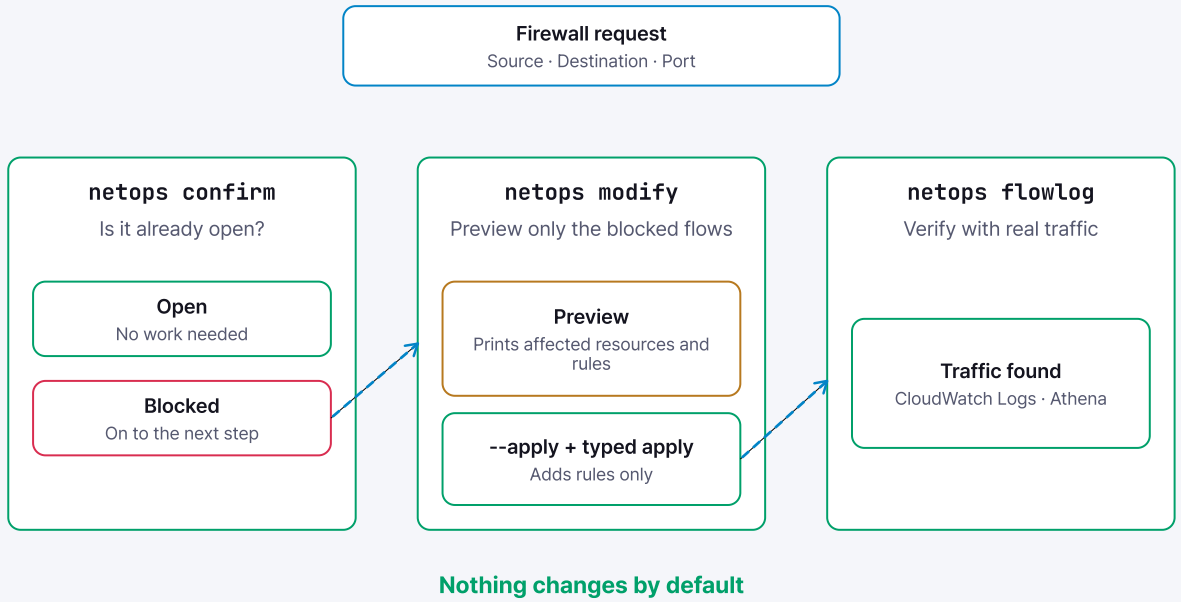
AWS firewall request automation

A CLI that checks whether a firewall request is already open, previews changes only for what is blocked, and verifies the real traffic.

3
Steps: confirm, modify, flowlog

4
Input formats

4
Resource types identified



Problem

Firewall requests often asked for flows that were already open, which added duplicate rules, and finding the related resources and rules meant searching the console account by account.

Approach

netops confirm resolves addresses to AWS resources and evaluates security group rules and routes. Only blocked flows go to modify, which prints a plan first. flowlog searches CloudWatch Logs or Athena for the actual traffic.

Result

By default nothing changes. Adding a rule needs --apply and a typed apply, rules can only be added, and a security group without free slots is never chosen.

Design and implementation · Personal project · released Nov 2026 · Python, boto3, AWS, CloudWatch Logs, Athena, Excel

github.com/kimjhtlzk/pf-09-aws-netops-firewall-automation

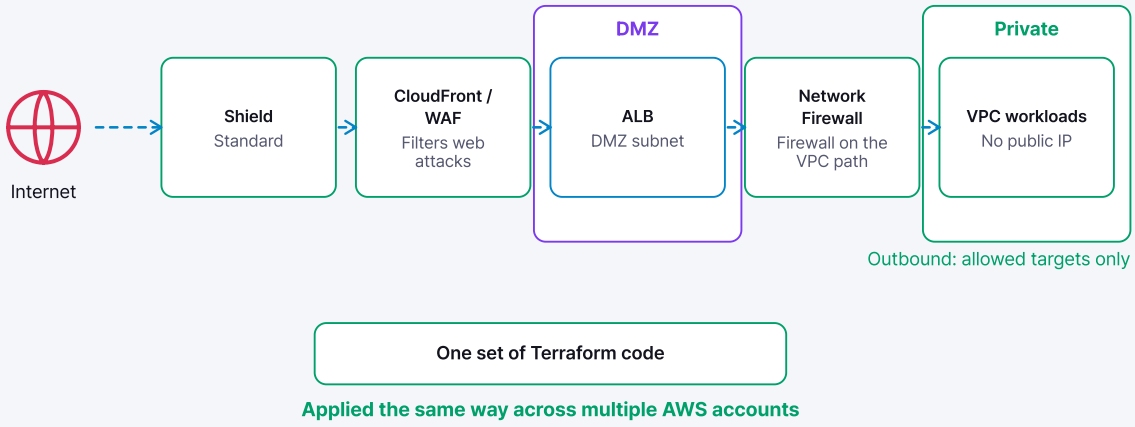
Woori Financial Group AWS security architecture

Moved network separation requirements across multiple AWS accounts into subnet and firewall structure, and standardized security settings as one set of Terraform code.

5
Security services connected to Splunk SIEM/SOAR

3
Design principles from network separation

1
Set of Terraform code across multiple AWS accounts



Problem

Group companies were adopting public cloud for the first time, with no Terraform in place, and security settings varied by account and by person. Network separation rules were hard to hold in documents alone.

Approach

Network separation (a Korean financial regulation) became three design principles: separate DMZ and Private subnets, no public IP on backends, and outbound only to allowed destinations. WAF, Network Firewall, and Shield Standard are defined in Terraform and applied through the same code to every account, with GuardDuty and Security Hub connected to Splunk SIEM/SOAR.

Result

The same security standard repeats through code as accounts grow, and per-person differences are gone. A Python pipeline now collects and analyzes network configuration across multiple AWS accounts, so review is repeatable. IAM permissions are defined in Terraform as well, which structurally blocks excessive grants.

Cloud Security Engineer · Network Security team, Information Security · Feb 2026 – Oct 2026 · Terraform, AWS Network Firewall, AWS WAF, AWS Shield Standard, GuardDuty, Security Hub, Splunk, Python

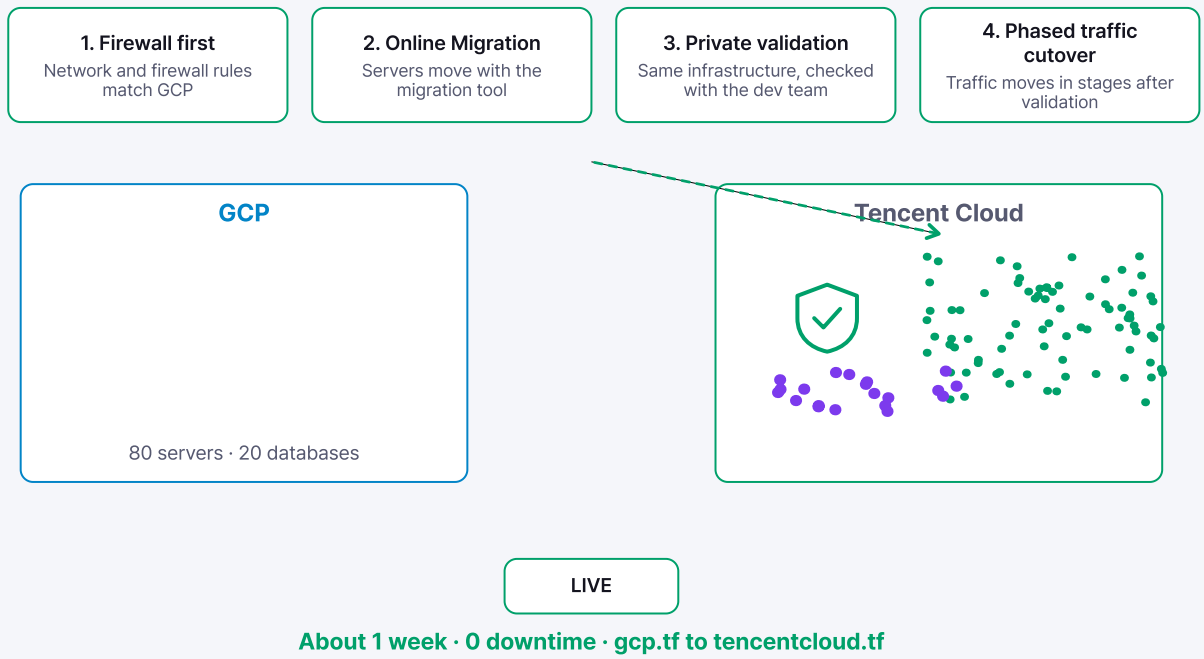
Zero-downtime migration from GCP to Tencent Cloud

Moved 80 servers and 20 databases of Summoners War: Chronicles from GCP to Tencent Cloud in about one week with no downtime.

80
Servers migrated

20
Databases migrated

0
Downtime



About 1 week · 0 downtime · gcp.tf to tencentcloud.tf

Problem

A live game had to move between clouds without stopping, and the firewall policy could not drift during the move.

Approach

Network and firewall rules were built on Tencent to match GCP first. Servers moved with the Online Migration tool and the same infrastructure was built. It then ran privately while the dev team checked for issues, and traffic moved in stages. The whole effort moved GCP Terraform code to Tencent Terraform code.

Result

The migration finished in about one week with no downtime. Some systems stayed on GCP while most game servers and databases moved to Tencent, and the firewall policy was fully reviewed and rebuilt along the way.

Cloud Engineer · Main engineer · Feb 2023 – Feb 2026 · Terraform, GCP, Tencent Cloud, Online Migration

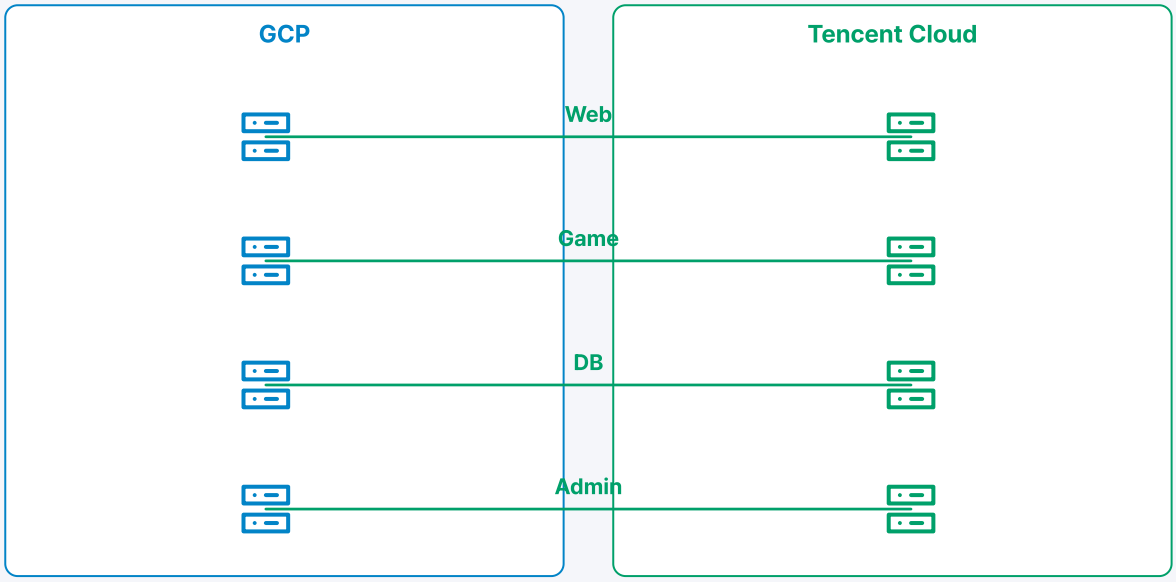
Summoners War: Sky Arena multi-cloud and firewall redesign

A massive game platform across GCP and Tencent Cloud, with 600 servers and 70 databases, a full firewall policy redesign, and patching with no maintenance window.

600
Servers

70
Databases

-40%
Deployment time after moving to Terragrunt



600 servers · 70 databases (across both clouds)

Problem

This was the company's largest project, with Tencent added for service in China. Firewall policies across the two clouds had grown duplicated and complex, and code spread across many regions had become hard to manage with Terraform alone.

Approach

A full analysis of cross-cloud firewall policies removed duplicate and unneeded rules and regrouped the rest by role: web, game, database, and admin. Terraform moved to a Terragrunt structure to remove code duplication. Most servers sat behind load balancers, so patches rolled through by taking servers out and back one at a time.

Result

Simpler firewall policies were easier to review and change, with a smaller exposed surface. Moving to Terragrunt cut deployment time by 40%, and GCP Cloud Armor provided DDoS protection.

Cloud Engineer · Main engineer · Feb 2023 – Feb 2026 · GCP, Tencent Cloud, Terraform, Terragrunt, Cloud Armor

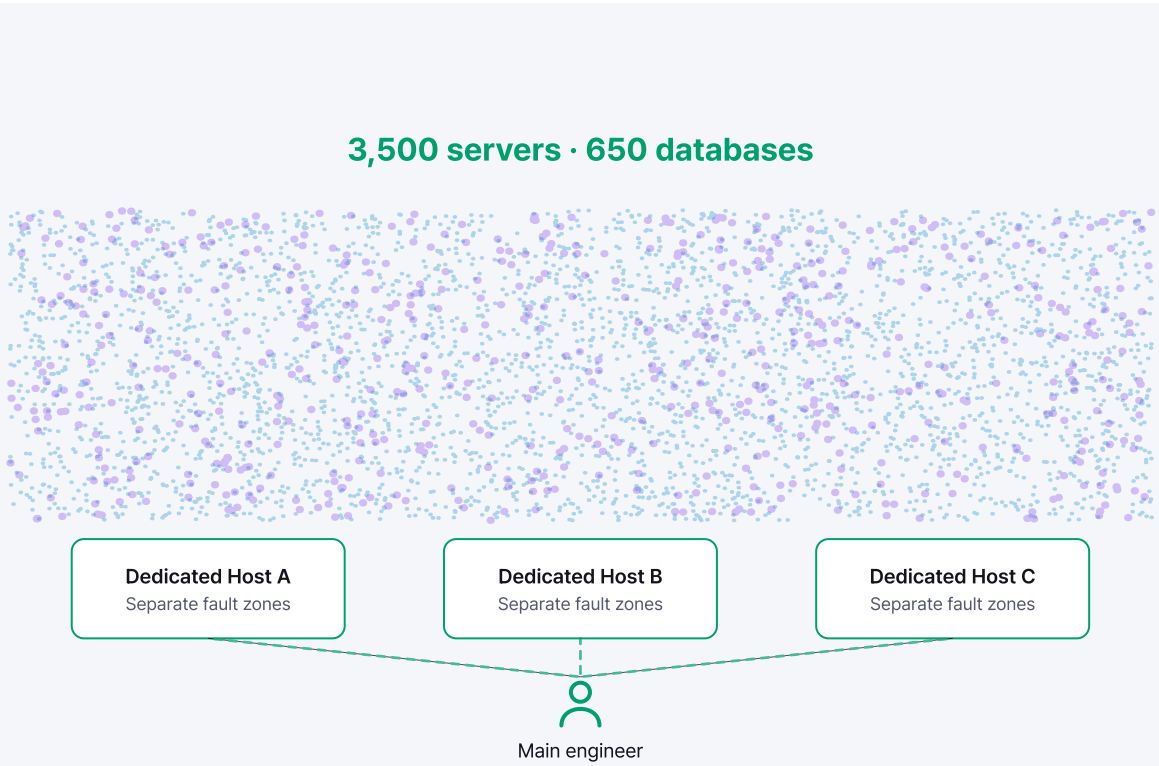
MIR4 Global: massive Azure infrastructure and automated DDoS blocking

Dedicated operation of a global game with 3,500 servers and 650 databases on Azure, plus a pipeline that finds DDoS sources in logs and adds them to an NSG automatically.

3,500
Servers

650
Databases

0
Human steps from detection to block



Problem

At this scale, a physical failure or an attack could spread across the global service. Night and weekend DDoS attacks were slow to answer when a person had to block them by hand.

Approach

Involved from the first design stage, I helped set a high-availability layout on Azure Dedicated Host. Mitigation flow logs from Azure DDoS protection are queried on a schedule in Log Analytics to find abnormal TCP SYN sources, and PowerShell adds those IPs to an NSG deny rule.

Result

Detection to block runs with no human step, which removed the delay at night and on weekends. Resource deployment, automated incident fixes, and asset collection were also automated in PowerShell.

Cloud Engineer · Main engineer (client: WEMADE) · Jul 2021 – Feb 2023 · Azure, Dedicated Host, PowerShell, Log Analytics, NSG

Dragon Sky FinOps: 62% lower infrastructure cost

Linked revenue metrics with infrastructure cost and found unused resources automatically every day with Python, cutting infrastructure cost by up to 62%.

-62%
Maximum infrastructure cost reduction

10
Servers

5
Databases



Problem

Infrastructure cost was high relative to game revenue, which hurt profitability.

Approach

Revenue metrics and infrastructure cost were monitored together. Python found unused resources across every cloud automatically each day.

Result

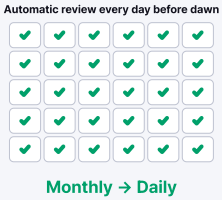
Infrastructure cost fell by up to 62%.

Cloud Engineer · Main engineer · Feb 2023 – Feb 2026 · Azure, Python, Billing monitoring

Field Notes

From a monthly security review to a daily automatic one

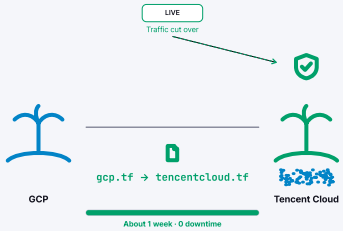
Monthly → Daily · 4 Clouds



Problem	Security reviews happened once a month and took days of manual work. If someone gave themselves excessive permissions or deleted a key server, we might not know for nearly a month. No one asked me to fix it. I decided to change it myself.
Judgment	Even a simple permission change was recorded differently in every cloud. I chose to convert every log to one common standard, "who did what, to which resource, and when".
What I did	Audit logs from every project in all four clouds are collected, normalized, and reviewed automatically before dawn each day. Only actions that could lead to an incident count as critical. They go to Telegram and Slack, and are recorded as resolved after the actor confirms.
Result	The review cycle went from monthly to daily, and awareness of risky actions went from up to a month later to the same day. Review effort dropped to effectively zero.
What I learned	Defining the signals that lead to incidents, and never missing them, matters more than analysis after the fact. Without authority, "I tried this and it got better" persuades more than "we should do this".

Moving a live game between clouds without stopping it

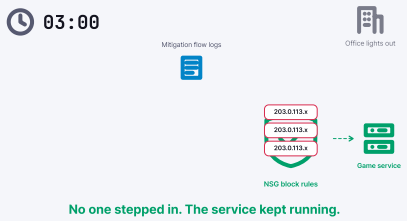
1 Week · 0 Downtime · 80 Servers / 20 DBs



Problem	A live service on GCP (80 servers, 20 databases) had to move to Tencent Cloud. Downtime was not acceptable.
Judgment	The biggest risk was security policy going missing or changing during the move. So I moved the network and firewall first, before any server.
What I did	Firewall rules were built on Tencent Cloud to match GCP first, then servers moved with Online Migration. Before cutover, the dev team and I checked for issues in private validation.
Result	After validation the service was cut over. The migration finished in about one week with zero downtime.
What I learned	When you move clouds, make the security standard move with you. The migration also stayed in Terraform code, which made later operations and changes easier.

Stopping 3 a.m. DDoS attacks with no one on the keyboard

`Detect → block automated · 0 human steps`



Problem	Attacks clustered at night and on weekends, when the fewest people were on hand. While someone read the alert, logged in, found the attacking IPs, and added block rules, the service shook.
Judgment	Instead of adding staff or buying a new product, I used logs that already existed. Azure's DDoS protection was already recording where attacks came from.
What I did	Sources sending abnormally many TCP SYN packets became attack candidates. Scheduled queries pull those IPs, and a PowerShell script adds them to NSG (firewall rule set) block rules automatically.
Result	Detection through blocking now runs with no human steps, which removed the night and weekend response delay. It needed no new product and no extra staff.
What I learned	Before adding a tool, see what the logs you already have can do. Stopping the attack is not enough. Build something that holds when it happens again.

Setting a security baseline for a financial cloud that had none

`IaC 0 → Terraform standard`



Problem	The group's companies were adopting public cloud for the first time, and there was no IaC (managing infrastructure as code) at all. Each account had different needs, but the common security baseline had to hold everywhere.
Judgment	I turned network separation (a Korean financial regulation that splits networks by sensitivity so outside threats cannot reach inside directly) into three design principles. Separate DMZ and Private subnets, no public IPs on backend resources, and outbound traffic to approved destinations only.
What I did	I introduced Terraform, defined Network Firewall and WAF as code, and applied the same code to every account.
Result	As accounts are added, the same security baseline repeats through code. Settings no longer vary by engineer.
What I learned	Put a rule in structure and code instead of a document, and every deployment meets the same bar.

Other field notes

Title	In one line	Key figure
Dedicated engineer for a 3,500-server global game	Dedicated engineer for a global MMORPG with 3,500 servers and 650 databases, covering Azure operations and technical advice.	3,500 Servers, 650 DBs
Deploying to 500 servers, 80% faster	Bundled deployment steps people repeated for every server group into PowerShell scripts, making deployments 80% faster.	80% faster, 500 Servers / 100 DBs
Infrastructure status at a glance, in real time	Built real-time dashboards with Telegraf, InfluxDB, and Grafana, and ran monitoring with Zabbix and Grafana.	
Making sure the same finding never comes back	Sorted security findings by service impact, agreed on compensating controls, and turned repeated findings into code defaults, raising the audit remediation rate from 60% to over 85%.	60% → 85%+
One codebase instead of copied infrastructure code	Moved live infrastructure into code and replaced copied Terraform with a shared Terragrunt structure, cutting deployment time 40%.	-40% deploy time
Redesigning a tangle of firewall rules	Reviewed every firewall rule across Tencent Cloud and GCP for 600 servers and 70 databases, then redesigned them around server roles.	600 Servers / 70 DBs
Stopping attacks on game servers	Ran DDoS detection and defense for a 600-server game on Tencent Cloud and GCP.	
Knowing first when something goes wrong	Watched server and database metrics with Zabbix and Grafana, responded to Slack and Telegram alerts, and analyzed monthly costs.	
An AI that writes infrastructure code from a plain request	Built an infrastructure code generation prototype with a local LLM and a vector database, demoed it to the team, and found its limits in testing. It did not reach day-to-day use.	
Only the access you need, recorded in code	I defined service account and role permissions in Terraform, so that granting too much access became structurally difficult.	
Alerts from many accounts on one screen	I connected GuardDuty, Security Hub, and Splunk SIEM to monitor security events across multiple AWS accounts in one place.	
"Turned on" and "actually blocking" are different things	In a CDN and load balancer setup, I found where a geo-block policy could be bypassed, fixed it, and proved the block with logs. This case is already fixed.	
Operations tools that run on a PC with no internet	I built an offline development environment and four Python tools for work PCs with no internet, turning repeated traffic requests into a standard process.	4 tools, 31 Python packages, bundled offline
Explaining in the language the other person already knows	I documented security group policies in the physical firewall format the team already knew, so engineers new to AWS could start work right away.	
Less time spent capturing packets	I automated FortiGate capture commands and pcapng file conversion with Python, which shortened the time needed to prepare an analysis.	
Owning the security checks that repeat every month	I operated firewalls, WAF, and WIPS, wrote the regular monthly review reports, and documented that work in handover notes when I left.	
A security review that tested requirements with numbers	I handled the security part of a review for adopting a new virtualization platform. I organized the requirements, tested them with numbers, and proposed a split of duties by traffic direction.	
A virtual company staffed by 11 AI employees	I built a virtual company of 11 AI employees on a hand-built orchestrator, and saw firsthand the trust boundary problem when agents accept each other's work without checks.	11 Agents, 3 Divisions
A service that answers "can these two talk?" in two clicks	I am turning the manual work of tracing AWS traffic paths in the console into a service where you pick two resources on a diagram and get an answer.	
A public repository series for this portfolio	I am rebuilding my existing GitHub practice and tool repositories as numbered public repositories and linking them to each experience.	

Production projects

Project	Cloud	Servers / DBs	Period
Summoners War: Sky Arena	GCP, Tencent	600 / 70	Feb 2023 – Feb 2026
MIR4 Global	Azure	3,500 / 650	Jul 2021 – Feb 2023
MIR M Korea	Azure	700 / 150	Feb 2022 – Feb 2023
MIR4 Korea	Azure	500 / 100	Dec 2020 – Feb 2023
The Legend of Mir 2/3 Korea	Azure	200 / 40	Dec 2020 – Feb 2023
Summoners War: Chronicles	GCP, Tencent	80 / 20	Feb 2023 – Feb 2026
Com2uS Pro Baseball V25	GCP	90 / 8	Feb 2023 – Feb 2026
Summoners War: Rush	Tencent	76 / 11	Mar 2024 – Feb 2026
Gods & Demons	GCP	66 / 12	Jul 2024 – Feb 2026
Starseed: Asnia Trigger	AWS	50 / 20	Sep 2023 – Feb 2026
MLB Rivals	GCP	23 / 5	Mar 2023 – Feb 2026
ADT-CAPS (SK Shieldus)	Azure	10 / 2	Mar 2021 – Feb 2023
DragonSky: Idle & Merge	Azure	10 / 5	Feb 2023 – Feb 2026
Summoners War: Lost Centuria	GCP	8 / 2	Feb 2023 – Feb 2026
Golf Star	GCP	6 / 2	Feb 2023 – Feb 2026
Ace Fishing: Crew	Tencent	4 / 2	Feb 2023 – Feb 2026
Idle Strikers 1945	Azure	4 / 1	Oct 2023 – Feb 2026
Strikers1945: RE	Tencent	3 / 2	Jun 2023 – Feb 2026
Minigame Party: Pocket Edition	GCP	3 / 4	Mar 2024 – Feb 2026

Education

Kyungmin University, Department of Information and Communication
Mar 2015 – Feb 2020 · Includes the military service period

Bachelor's degree, Academic Credit Bank System (National Institute for Lifelong Education)
Dec 2019 – Aug 2020 · Degree conferred in Information and Communication

KG ITBANK: AWS Architecture Design Specialist Training Program
Feb 2020 – Jul 2020 · 6 months, 800 hours. Project: migrating a fictional client to AWS

Military service (Republic of Korea Army, honorably discharged)
Dec 2016 – Sep 2018 · Sergeant. Signal soldier (switchboard operator)

Certifications

NCP — Naver Cloud Platform Certified Professional
2021.08.18

NCA — Naver Cloud Platform Certified Associate
2021.07.07

Engineer Information Processing (Korean national certification)
2021.06.02

Microsoft Azure Fundamentals (AZ-900)
2020.10.22

Linux Master Level 2
2019.10.11

Network Administrator Level 2
2019.07.09

HashiCorp Certified: Terraform Associate
2022.12.18 · History

CKA — Certified Kubernetes Administrator
2022.12.16 · History

Microsoft Azure Database Administrator Associate
2022.06.09 · History

Microsoft Azure Solutions Architect Expert
2022.06.08 · History

Microsoft Azure Administrator Associate
2022.06.07 · History

AWS Certified Solutions Architect – Professional
2022.05.29 · History

Microsoft Azure Architect Technologies
2021.07.01 · History

Microsoft Azure Security Engineer Associate
2021.06.18 · History

AWS Certified Solutions Architect – Associate
2020.06.23 · History

Principles from the work

> **Before adding a tool, see what the logs you already have can do.**
>
> Stopping 3 a.m. DDoS attacks with no one on the keyboard

> **Without authority, "I tried this and it got better" persuades more than "we should do this".**
>
> From a monthly security review to a daily automatic one

> **Stop at "no" and the conversation ends. Bring an alternative and things get fixed.**
>
> Making sure the same finding never comes back

> **When you move clouds, make the security standard move with you.**
>
> Moving a live game between clouds without stopping it

> **Put the rule in structure and code, and every deployment meets the same bar.**
>
> Setting a security baseline for a financial cloud that had none

> **Turning a control on is different from proving it stops the threat.**
>
> "Turned on" and "actually blocking" are different things

> **The requested fix ("please open it") and the real problem can differ. Check what is failing first.**
>
> Operations tools that run on a PC with no internet

> **AI supports the judgment. A person approves the action.**
>
> A virtual company staffed by 11 AI employees