Path	Question	Answer	Note						
CICD	1. Mô tả pipeline CI/CD đã triển khai:	**- Pipeline bao gồm:** 1.Code checkout (Git) 2. Static Code Analysis (SAST) <-- **DevSecOps** 3. Build image (Docker) 4. Image Vulnerability Scan <-- **DevSecOps** 5. Test unit/integration 6. Secret Scan <-- **DevSecOps** 7. Push image (Docker Registry) 8. IaC Security Scan <-- **DevSecOps** 9. Deploy to staging/prod (Helm/Kubectl). 10. Runtime Security / Monitoring <-- **DevSecOps** **- Dùng GitOps (ArgoCD) cho production.**	**2. Static Code Analysis (SAST)** **Mục tiêu:** Phát hiện lỗi bảo mật trong code trước khi build. **Công cụ:** + Python: bandit + JavaScript: eslint, npm audit + Java: SonarQube, FindSecBugs	**4. Image Vulnerability Scanning** **Mục tiêu:** Quét lỗ hổng bảo mật trong Docker image. **Công cụ:** + Trivy + Grype + Anchore + Clair + Snyk	**5. Unit Test** **Mục tiêu:** Kiểm tra tính đúng đắn của các hàm/mô-đun riêng lẻ. **Thực hiện tại đâu:** Trong job CI đầu tiên sau khi build hoặc trong cùng container build. **Công cụ thường dùng (tùy ngôn ngữ):** + Python: pytest, unittest + Node.js: jest, mocha + Java: JUnit, Mockito + Go: go test	**5. Integration Test** **Mục tiêu:** Kiểm tra sự tương tác giữa các module hoặc với dịch vụ bên ngoài như DB, API. **Môi trường:** + Dùng Docker Compose để spin up service cần thiết (DB, Redis, ...). + Hoặc deploy tạm lên staging namespace trong Kubernetes. **Công cụ**: + Postman/Newman, pytest (Python), supertest (Node.js) + Kết hợp với docker-compose, minikube, hoặc kind.	**6. Secret Scan (Hardcoded Secrets)** **Mục tiêu:** Phát hiện key/token/password bị push vào repo hoặc image. **Công cụ:** + gitleaks + detect-secrets + trufflehog	**8. IaC Security Scan** **Mục tiêu:** Quét file Terraform, Helm, YAML để phát hiện cấu hình sai. **Công cụ:** + Checkov + tfsec + kube-score	**10. Runtime Security** **Công cụ triển khai sau khi deploy production:** + Falco + Sysdig + Aqua + Datadog CSPM Alert nếu container chạy shell bất thường, mount volume trái phép, v.v.
	2. Công cụ CI/CD đã dùng:	- Jenkins, GitLab CI, GitHub Actions. - Dễu linh hoạt, dễ tuỳ chỉnh. - Chọn tuỳ theo nhu cầu dự án: + Jenkins cho dự án lớn. + GitHub Actions cho open-source + GitLab CI cho private repo.							
	3. Zero-downtime deployment:	- Dùng Rolling update (K8s), Blue/Green deployment hoặc Canary release. - Load balancer tự điều hướng traffic dần dần sang version mới.							
	4. Rollback:	- Trong K8s: `kubectl rollout undo deployment`. - Trong Helm: `helm rollback <release> <revision>`. - Triển khai đa staging/test để giảm nguy cơ lỗi production.							
	5. Secret trong pipeline:	- Lưu trong Secret Manager (AWS Secrets Manager, Vault, GitLab CI Variables). - Inject qua environment variables hoặc mount đúng scope. - Tránh log lộ secrets, đặt mask trong CI tool.							
Cloud Platforms	1. Kiến trúc cloud gần nhất:	- Sử dụng AWS: ALB → ECS (Fargate) → AuroraDB → S3 → CloudFront. - CI/CD bằng CodePipeline + CodeDeploy. Monitoring bằng CloudWatch + Datadog.							
	2. So sánh AWS vs GCP:	- EC2 vs GCE: Tương đồng, GCE startup nhanh hơn. - S3 vs GCS: Tương tự, GCS thân thiện với gsutil. - IAM: AWS phức tạp hơn, nhưng tính năng rõ ràng hơn.							
	3. Terraform vs CloudFormation:	- Terraform: Multi-cloud, declarative, reusable modules, debug dễ. - CloudFormation: Native AWS, tích hợp sâu, chạy chậm hơn, syntax khó hơn.							
Containerization (Docker, Kubernetes)	1. Triển khai microservices với K8s:	- Mỗi service được package Docker image → deploy với Helm chart. - Dùng Ingress Controller + Service Mesh (Istio) cho routing, auth. - ConfigMap & Secret cho config separation.							
	2. Deployment vs StatefulSet:	**Deployment** **- Dùng cho loại ứng dụng:** Stateless (web, API, frontend...) **- Định danh Pod:** Không cố định, pod mới có tên mới **- Volume (Persistent):** Volume có thể chia sẻ hoặc bị tạo lại **- Quản lý trạng thái (state):** Không lưu trạng thái theo từng pod **- Truy cập mạng:** Dùng Service IP, không cần DNS riêng từng pod **- Cập nhật:** Rolling update theo kiểu hỗ trợ **- Khả năng scale out/in:** Scale nhanh, không lo mất dữ liệu **- Use case phổ biến:** Frontend, stateless API, worker job	**StatefulSet** **- Dùng cho loại ứng dụng:** Stateful (DB như MySQL, Kafka, Cassandra, Zookeeper...) **- Định danh Pod:** Tên pod cố định theo thứ tự (pod-0, pod-1, ...) **- Volume (Persistent):** Volume gắn liền với pod theo chỉ số (index) cụ thể **- Quản lý trạng thái (state):** Duy trì trạng thái theo từng pod riêng biệt **- Truy cập mạng:** Mỗi pod có DNS riêng (pod-0.svc-name.namespace.svc.cluster.local) **- Cập nhật:** Cập nhật tuần tự từ pod-0 → pod-n **- Khả năng scale out/in:** Scale phải cẩn thận vì có thể ảnh hưởng đến dữ liệu **- Use case phổ biến:** Database, Kafka, Elasticsearch, Redis Cluster...					
	3. Monitor K8s cluster:	- Prometheus + Grafana. - Metrics-server, kube-state-metrics. - Alert qua Alertmanager, Slack/email.							
	4. CrashLoopBackOff:	- Kiểm tra log pod (`kubectl logs`). - Kiểm tra livenessProbe, readinessProbe. - Kiểm tra image tag, secret/config mount, dependency không tồn tại.							
Bảo Mật	1. Biện pháp bảo mật DevOps:	- Sử dụng IAM, least privilege. - Rotate secret định kì. - Quản lý log & audit. - Network segmentation (VPC, SG).	**Sử dụng IAM, least privilege** - Chỉ cấp quyền tối thiểu cần thiết cho từng user/service. - Tránh gán quyền "AdministratorAccess" không cần thiết. **Ví dụ:** Dev chỉ cần quyền **s3:PutObject**, không cần **s3:DeleteBucket**.	**Rotate secret định kì** - Thay đổi (rotate) API keys, DB passwords định kỳ để hạn chế rủi ro khi bị lộ. **Công cụ hỗ trợ:** AWS Secrets Manager, Vault, Doppler.	**Quản lý log & audit** - Ghi lại mọi hành động (access log, change log, pipeline log) để điều tra nếu có sự cố. - Dùng CloudTrail (AWS), Audit Logs (GCP), ELK stack.	**Network segmentation** - Tách biệt các thành phần bằng VPC, subnet, Security Groups, NACLs. **Ví dụ:** Frontend không được truy cập DB trực tiếp, chỉ qua backend API.			
	2. Credential trong pipeline:	- Lưu trong secret store. - Mask ở CI/CD tool. - Không hardcode trong repo.	**Lưu trong Secret Store** Không để mật khẩu/API key trong code. **Sử dụng:** GitHub Actions secrets, GitLab CI secret variables, Vault, AWS Secrets Manager.	**Mask output ở CI/CD** Đảm bảo log pipeline không hiển thị giá trị secret. **Ví dụ:** *** thay vì in ra token thực tế.	**Không hardcode vào repo** Tuyệt đối không commit credentials, token, API key trong mã nguồn (dù là file .env). **Công cụ phát hiện:** gitleaks, truffleHog.				
	3. IAM roles/policies AWS:	- Tách nhóm theo vai trò. - Nguyên tắc principle of least privilege. - Gán IAM role cho service thay vì access key.	**Tách nhóm theo vai trò** Gán user/group theo vai trò: Dev, QA, Ops... → dễ kiểm soát permission & audit. **Ví dụ:** nhóm dev-readonly-s3, qa-admin-eks	**Principle of Least Privilege** Tạo policy chỉ cấp quyền cần thiết, giới hạn theo điều kiện (resource, IP, time, ...). **Ví dụ:** chỉ cho phép xóa S3 bucket từ IP công ty.	**Gán IAM Role cho service** - Không dùng access key trong ứng dụng. - EC2/ECS/Lambda có thể assume role để gọi dịch vụ AWS. **Ví dụ:** EC2 gán role s3-readonly-role để đọc file từ S3 mà không cần key.				
Monitoring & Logging	1. Stack logging/monitoring:	- Prometheus + Grafana cho metric. - Vector/Loki/ELK cho log. - Datadog/CloudWatch cho alert nhanh.							
	2. Alert khi sự cố:	- Dùng Alertmanager, PagerDuty, Slack webhook. - Thiết lập threshold + anomaly detection.							
	3. Xác định root cause:	- Correlate log, metric và trace. - Dùng tracing tool: Jaeger, OpenTelemetry. - Kiểm tra thay đổi gần đây (Git, CI/CD).							
Troubleshooting	1. App không start trong container:	- `docker logs`, `kubectl logs`, `describe pod`. - Kiểm tra ENTRYPOINT, env vars, secret mount. - Chạy local với `docker run -it` để debug.							
	2. CI/CD pipeline chậm:	- Xem step nào chậm nhất. - Kiểm tra caching (Docker layer, dependency). - Tối ưu lại Dockerfile, split job.							
	3. Debug network-level trong hệ thống phân tán:	- Dùng `tcpdump`, `traceroute`, `curl`, `netcat`. - Kiểm tra DNS resolution, firewall rule, security group. - Dùng network policy (K8s) để trace path.							
Scripting & Automation	1. Ngôn ngữ scripting:	**Bash** **Ưu điểm:** - Nhẹ, có sẵn trong hầu hết Linux distro - Phù hợp với task đơn giản (copy file, chạy lệnh CLI, SSH…). **Khi dùng:** - Viết job cho Jenkins/GitLab runner. - Tối ưu lại Dockerfile, split job.	**Python** **Ưu điểm:** - Dễ đọc, mạnh mẽ, hỗ trợ REST API rất tốt. - Nhiều thư viện: requests, boto3, paramiko. **Khi dùng:** - Kết nối cloud APIs (AWS, GCP). - Phân tích log, backup dữ liệu, kiểm tra sức khỏe hệ thống.	**Go** **Ưu điểm:** - Hiệu suất cao, dễ biên dịch ra binary. - Quản lý đồng thời (concurrency) tốt. **Khi dùng:** - Dùng khi cần chạy tool tự động tốc độ cao. - Tool CLI nội bộ hoặc daemon giám sát.					
	2. Mô tả script automation đã viết:	**Deploy + Rollback microservice bằng Helm** **Mục đích & Chi tiết:** - Viết shell script nhận version, env, release name. - Dùng helm upgrade để deploy, rollback nếu kubectl rollout status fail. **Điểm hay:** tích hợp với Slack để báo kết quả.	**Tự động backup database và upload S3** **Mục đích & Chi tiết:** - Viết Python script chạy theo cron: → Dump DB → gzip → upload lên S3 bằng boto3. - Gửi email khi backup thành công/thất bại.	**Script kiểm tra health API để quyết định rollout** **Mục đích & Chi tiết:** - Trước khi deploy version mới: → Call API /healthz ở staging, nếu trả về 200 → continue rollout. → Nếu fail → rollback hoặc stop pipeline.					
	3. Đánh giá script "tốt":	**Idempotent** Dù chạy bao nhiêu lần cũng không gây hại (ví dụ: không tạo duplicate file/user) **Logging rõ ràng** Log ra console + file (nếu cần), phân loại lỗi/thành công **Retry khi thất bại** Gặp lỗi tạm thời (API timeout, DB lock) thì retry vài lần trước khi fail	**Tham số hoá (parametrize)** Không hardcode config → truyền qua biến môi trường, flag CLI, hoặc file YAML **Test được / có cấu trúc rõ ràng** Tách function, dễ test, dễ đọc lại sau này **Không để lộ thông tin nhạy cảm** Không log token/password, che giấu đầu ra nếu cần						
Lãnh đạo & Teamwork	1. Truyền đạt DevOps cho team non-tech:	- Ví dụ đời thời, dễ hiểu. - Demo CI/CD flow. - Tổ chức workshop nội bộ.							
	2. Mentoring junior DevOps:	- Code review, pair programming. - Giao bài thực tế, chia nhỏ. - Định hướng mindset: logging, reproducibility, automation.							
	3. Giải quyết Dev vs Ops conflict:	- Hiểu nhu cầu của đôi bên. - Thiết lập SLO/SLA chung. - Tạo shared ownership: "you build it, you run it".							

Tôi là Vũ Hồng Phúc, hiện đang là DevOps Engineer tại CMC Global với hơn 5 năm kinh nghiệm trong lĩnh vực vận hành hệ thống và phát triển hạ tầng tự động hoá.

Tôi có kinh nghiệm triển khai và vận hành các hệ thống Microservices trên nền tảng Kubernetes cả On-premise và Cloud (AWS, Samsung Cloud), đồng thời thành thạo các công cụ CI/CD như Jenkins, GitLab CI và ArgoCD (GitOps).

Bên cạnh đó, tôi còn có kỹ năng vững vàng với các công cụ giám sát hệ thống như Grafana, Prometheus, ELK Stack, Vector cùng khả năng scripting bằng Python và Bash để tối ưu hóa quy trình làm việc.

Tôi có định hướng trở thành một DevOps Leader trong tương lai, sẵn sàng chia sẻ kiến thức, hỗ trợ đồng đội và nâng cao trình độ chuyên môn thông qua các dự án thực tế.

Tôi cũng đã đạt các chứng chỉ quốc tế như CKA và CKAD, phản ánh cam kết lâu dài với lĩnh vực DevOps và Cloud Native.