GrowBit HR System

PERSONAL DEVELOPMENT PLAN Personal Development Plan (PDP)

Employee: Berik Bazarbayev

Position: DevOps Engineer

Introduction

This Personal Development Plan is designed to guide Berik's professional growth over the next 12 months, aligning his individual goals with the needs of the team and the company. The plan considers his strong practical skills in CI/CD, release security, and operational procedures, as well as identified gaps in his fundamental theoretical knowledge of infrastructure and distributed systems.

Goal

The goal is to preserve and enhance his current strengths—a pragmatic approach to automation, backups, and rollbacks, as well as a mature approach to monitoring and on-call—while closing critical gaps that could affect the reliability and security of Fintech products. The plan combines technical training, practical tasks, mentorship, and progress metrics to make the development concrete and measurable. Executing this plan will help Berik increase his confidence in architectural decisions, mitigate security risks, and become a more valuable team member and a potential future leader.

Key Development Goals

1. **Strengthen fundamental knowledge of distributed systems and infrastructure principles.** This is critical for making sound architectural decisions (CAP theorem, immutable infrastructure) and reducing risks during service scaling.
2. **Correct specific misunderstandings of Kubernetes** (the role of kube-scheduler, Pod scheduling, session affinity). Correct knowledge will enable more reliable orchestration design and debugging.
3. **Improve monitoring and metric practices** (including p99 latency and the SLI/SLO approach). This is important for objectively evaluating user latency and reducing incidents.
4. **Enhance the security of secrets and access management** (centralized storage, key rotation, revocation). This is a necessary goal to comply with Fintech standards and reduce the risk of leaks.
5. **Consolidate and formalize strengths in CI/CD:** automated security checks, rollbacks, and release control. This will increase the speed and security of delivery.
6. **Gain practical experience in incident response and on-call scenarios** (roles, checklists, communication). This will improve MTTR and the quality of postmortems.
7. **Develop communication and documentation skills** so that technical solutions are understandable to the team and easily reproducible.
8. **Prepare for a mentor/technical leader role at the team level:** sharing knowledge on secure release and monitoring practices.

Main Development Areas

1. Foundational Principles of Distributed Systems An understanding of the principles of distributed systems (CAP theorem, consistency vs. availability, network partitions) is the foundation for designing resilient services. Developing competence in this area will allow Berik to make informed trade-offs when designing data and replication, avoiding incorrect analogies and errors that were identified in the test. Skills will include reading architectural patterns, analyzing failure scenarios, and applying correct replication and consistency strategies in a Fintech context. Mastering these concepts will reduce the likelihood of design errors that require costly fixes in production. It will also strengthen his ability to make architectural trade-offs and justify decisions to the team and management.

2. Kubernetes and Container Orchestration Precise knowledge of Kubernetes components (kube-scheduler, kube-controller-manager, etc.), Pod scheduling models, and session affinity mechanics is critical for correctly distributing loads and ensuring SLAs. Working on this area involves practical exercises on tuning the scheduler, profiling Pods, testing affinity policies, and understanding the Pod lifecycle. Competencies include writing manifests with clear anti-affinity rules, using node selectors, taints/tolerations, and session affinity mechanisms. This will reduce the risks of incorrect load distribution and unintended downtime during updates.

---

3. Observability and Metrics Shifting from "number of connections" to an SLI/SLO approach with p50/p95/p99 latency metrics will provide an objective picture of user latency. Developing these competencies includes choosing the right metrics for business-critical paths, setting up distributed tracing, and creating functional dashboards and alerts that minimize false positives. Berik will learn how to configure the export of latency metrics, build correct alerting rules, and analyze latency distributions. This will improve the quality of incident investigations and allow a focus on problems that directly affect users.

4. Secrets Security and Access Management The test revealed dangerous practices (suggesting to send secrets via email) and incomplete approaches to key revocation/rotation. The goal here is to master centralized secret stores (Vault, Secrets Manager) with rotation policies, limited privileges, and automated distribution. It is important to implement secret-scanning practices in CI and procedures for rapid key revocation with a minimal blast radius. Competencies will include mastering infrastructure for secret storage, integration with CI/CD, and planning secure procedures for restoring trust after leaks.

5. CI/CD, Secure Releases, and Rollback Practices Berik already demonstrates a pragmatic approach to CI/CD: automated security checks and rollbacks. These practices need to be formalized and expanded: implementing testing channels, canary releases, automated rollbacks based on metrics, and ensuring SAST/DAST integration. Development in this area will enhance the ability to conduct fast and secure releases, reduce the time spent on manual checks, and decrease the risk of regressions in production.

6. Incident Response, On-Call, and Operational Discipline An understanding of the sequential steps in a secret leak and recovery plans is a strong point that needs to be further developed into practical scenarios. Skills will include developing playbooks, conducting incident drills, improving communication in emergency cases, and fostering a postmortem culture. This will increase the team's response speed and reduce the number of repeated incidents.

12-Month Action Plan (6 phases, 2 months each)

Phase 1 (Months 1-2)

Task Description: Begin by aligning fundamental knowledge and correcting identified misconceptions. The goal is to close the most critical conceptual gaps (CAP theorem, immutable infrastructure, the role of kube-scheduler) and create a plan for practical exercises. Simultaneously, audit the current CI/CD pipelines and secret management practices to establish a baseline for improvements. The focus is on theory with short practical tasks to build a foundation for subsequent deep practices.

Detailed Actions:

- Study key concepts of distributed systems: read chapters/content on CAP, consistency, and resilience ; complete short quizzes after each topic to solidify understanding. This will address fundamental design errors.
- Go through a practical guide to the Kubernetes scheduler: reproduce Pod scheduling scenarios in a test environment, change scheduler configurations, and observe behavior. This practice will reinforce theory and eliminate incorrect assumptions.
- Conduct a mini-audit of CI/CD and secrets: list the tools and processes used, identify where secrets are stored/transferred, and describe current risk points. This will provide a concrete list of prioritized improvements.

  Expected Outcome: By the end of this phase, Berik will be able to confidently explain key distributed systems concepts and the role of basic Kubernetes components. A preliminary checklist of problems in CI/CD and secret management will be created, with priorities for correction. A plan for practical exercises and a list of metrics to start tracking in subsequent phases will be established.

  Additional Resources: Intensive reading and practical materials are recommended: the CAP chapter from a book on distributed systems and the official Kubernetes scheduler documentation for practice. Use online sandboxes (minikube/k3s) to reproduce scheduling scenarios. For the CI/CD audit, export pipeline configurations and create a simple risk point table. Internal meetings with a senior DevOps/architect to discuss identified gaps will help convert theory into practice. Major articles/white papers on immutable infrastructure will help reinforce the stance against manual edits. Small tests/quizzes after reading will solidify the material and reveal any remaining gaps.

Phase 2 (Months 3-4)

Task Description: Focus on the practical mastery of Kubernetes and the implementation of correct session affinity and load planning patterns. Simultaneously, begin implementing a basic secret management system in a test environment. The goal is to turn knowledge into reproducible processes and a minimally viable secure workflow for secrets and deployments.

Detailed Actions:

- Set up a test cluster and implement examples of affinity/anti-affinity, taints/tolerations, and node selectors; document the results and risks of each approach. This will provide practical experience in choosing a placement strategy.
- Implement a simple secret storage system in the test environment (e.g., HashiCorp Vault or equivalent) and integrate it with CI for automated access without email distribution. This will eliminate a basic vulnerability in current practices.

- Conduct a peer review of configurations with a senior engineer and document the changes in the internal wiki. Feedback will accelerate the correction of approaches and provide standardized documentation.

  Expected Outcome: By the end of this phase, there will be a working test cluster with correctly configured placement patterns and detailed documentation on the strategies used. A basic secure secret store and CI integration for the test environment will be implemented. The most obvious risks (e.g., sending secrets via email) will be eliminated.

  Additional Resources: Practical guides to the Kubernetes scheduler and official Kubernetes patterns; hands-on labs with k3s/minikube. Documentation and quick-start guides for HashiCorp Vault or a cloud provider's secrets manager for quick integration. Internal paired sessions with a senior DevOps engineer for reviews and discussion of trade-offs. Checklists for secret security and standard operating procedures (SOP) for rolling back and updating secrets are recommended. It is also recommended to keep a change log to track the results of experiments and errors.

Phase 3 (Months 5-6)

Task Description: Deepen observability practices: implement the collection of latency metrics (p50/p95/p99), tracing, and the creation of SLIs/SLOs for critical services. Concurrently, formalize CI/CD practices: canary releases, automated rollbacks based on metrics, and SAST/DAST integration. The goal is to make releases safe and measurable from a user experience perspective.

Detailed Actions:

- Configure the collection of p99 latency for key HTTP/DB paths and add distributed tracing (OpenTelemetry) to test services. Build dashboards and alerts based on p95/p99. This will capture real user latency.
- Implement a canary release in a test environment with automated metric analysis and rollback rules in case of degradation. This will reduce the risk of widespread impact from problems.
- Integrate SAST/DAST checks into CI and establish policies for mandatory security checks before merging. This will strengthen release security at an early stage.

  Expected Outcome: By the end of this phase, working dashboards with p99 metrics will be available, along with automated checks for releases and a canary release process with automatic rollback rules. The release process will become safer and more manageable, with user latency analytics as the basis for release decisions.

  Additional Resources: Books and materials on observability: articles and practical guides on OpenTelemetry, Prometheus, and Grafana ; examples of SLI/SLOs from SRE practices. Courses on building distributed request tracing and hands-on labs. Documentation on canary releases and feature flags (e.g., a library for feature flags). Regular code reviews for security checks and external reviews of metric configurations are recommended.

Phase 4 (Months 7-8)

Task Description: Focus on practicing incident procedures: write playbooks for common incidents (secret leak, latency degradation, scheduler malfunction), and conduct simulations and on-call drills.

Simultaneously, scale up secret management practices and automate key rotation. The goal is to reduce MTTR and formalize communication during incidents.

Detailed Actions:

- Develop a set of playbooks with clear steps for typical incidents, including communication templates and steps for revoking/reissuing keys. This will ensure a quick and coordinated response.
- Conduct at least one incident simulation (game day) with a real on-call rotation and a subsequent postmortem. The practice will reveal organizational gaps and provide experience in stressful communication situations.
- Implement automatic key/secret rotation for critical systems and test rollbacks without downtime. This will reduce the blast radius in case of a compromise.

  Expected Outcome: By the end of the phase, playbook templates will be ready, and at least one incident simulation will have been conducted with documented conclusions. Key rotation will be automated for critical components, and MTTR during simulations will decrease.

  Additional Resources: Materials on incident management and postmortem culture (SRE approaches). Practical cases on key revocation and rotation, communication checklists for incidents. Internal training and retrospectives after game days. Tools for secret orchestration and examples of CI/CD integration.

Phase 5 (Months 9-10)

Task Description: Strengthen leadership and communication skills through knowledge sharing: Berik should conduct masterclasses on CI/CD security, monitoring, and secret management for the team. Concurrently, he should participate in architectural discussions and propose well-reasoned trade-offs for system design. The goal is to increase Berik's influence within the team and solidify his mentorship skills.

Detailed Actions:

- Prepare and conduct at least two internal workshops: one on secure CI/CD and rollback practices, and a second on choosing the right metrics and interpreting p99. This will solidify his knowledge and demonstrate critical expertise.
- Take on the role of a technical owner for a small project or an observability improvement initiative, managing documentation and architectural decisions. Leadership practice in small projects is safe and effective.
- Engage in paired programming/reviews with less experienced colleagues, focusing on security and reliability. This will accelerate the team's growth and strengthen Berik's communication skills.

  Expected Outcome: By the end of the phase, Berik will have conducted training sessions that raise the overall level of the team and will have established himself as a practical mentor. His participation in architectural discussions will become noticeable, and his proposed solutions will be based on measurements and practical experiments.

  Additional Resources: Materials on effective presentation of technical topics, internal templates for workshops, and preparation checklists. Books on DevOps/DevSecOps and SRE that can be used as a basis for workshop content. Mentorship with a senior engineer for preparation and feedback on training formats.

Phase 6 (Months 11-12)

Task Description: The final stage focuses on consolidating results: formalizing processes (SOPs), final testing of real scenarios (a full-scale game day), and evaluating progress against KPIs. Prepare a final report on the progress achieved and propose a roadmap for the next year. The goal is to obtain objective evidence of improvements and define further steps.

Detailed Actions:

- Conduct a major game day simulating a complex incident (e.g., a secret leak + latency degradation) with a full cycle of response measures, including communication and investigation. This will test the maturity of the processes.
- Formalize a set of SOPs and internal policies for CI/CD, secrets, monitoring, and incidents, approved by management. These documents will ensure the reproducibility of best practices.
- Prepare a final self-assessment and presentation for the manager with progress metrics and development proposals for the next year. This will create transparency and a foundation for career development.

  Expected Outcome: By the end of the 12th month, processes will be formalized, a major incident simulation will have been completed with documented improvements, and a progress report will be prepared. Metrics such as deployment frequency, change failure rate, MTTR, and p99 latency for key paths will show improvement compared to the initial state. Berik will gain recognition for strengthening operations and will be ready for the next career steps.

Additional Resources (General Recommendations)

Recommended books and materials that directly address the identified gaps and goals:

- ***Designing Data-Intensive Applications*** (Martin Kleppmann) helps to understand CAP, consistency, and data handling patterns in distributed systems.
- Official Kubernetes documentation and the chapter on the scheduler (kubernetes.io) +

  Kubernetes Up & Running for a practical understanding of component roles and Pod scheduling.

- ***Site Reliability Engineering*** (Google SRE) and articles on SLIs/SLOs for developing observability and p99 metrics.
- HashiCorp Vault docs / cloud provider secrets manager guides: practical guides for secure storage and rotation of secrets.
- ***The DevOps Handbook*** and materials on canary releases and feature flags for systematizing CI/CD practices.
- Practical labs on OpenTelemetry, Prometheus, and Grafana (online courses or internal sandboxes) for practicing tracing and monitoring.
- Internal mentorship and paired sessions with senior engineers, as well as participation in code/infra reviews, are key practices for solidifying knowledge.

  Each resource directly corresponds to the identified gaps and will help transform theory into applicable commands and processes.

Measuring Progress

Progress will be evaluated by a combination of quantitative and qualitative methods:

- **Monthly one-on-ones with the manager** to discuss tasks, blockers, and goal achievement; these meetings will provide regular course correction.
- **Quarterly technical assessments:** small practical tests/demonstrations (reproduction scenarios) on topics like Kubernetes, distributed systems, and secrets. Specific tasks will show the level of assimilation.
- **360-degree feedback from colleagues (peer reviews)** after workshops and paired sessions to evaluate communication and mentorship skills.
- **Operational metrics:** deployment frequency, change failure rate, MTTR, and p99 latency for key paths. These KPIs will show the real impact of the improvements on the business.
- **Results of incident simulations (game day)** and subsequent postmortems: speed of playbook execution, completeness of reports, and implementation of recommendations.

  This multi-channel approach will allow for the assessment of both theoretical knowledge and the ability to apply it in production conditions, which is critical for a DevOps role in Fintech.

Vision After 12 Months

In one year, Berik will become a confident operations engineer with a solid theoretical foundation in distributed systems and practical skills confirmed by objective metrics. He will precisely understand the role of the kube-scheduler and correctly apply Pod scheduling patterns, which will reduce orchestration errors. His approach to observability will be based on SLI/SLO and p99 metrics, allowing for a more accurate assessment of user latency and informed release decisions. The secret management system and rotation procedures will be implemented and debugged, minimizing the risk of leaks and ensuring a quick response. CI/CD pipelines will be formalized with canary releases and automated rollbacks, and SAST/DAST integration will reduce the number of vulnerabilities before release. Berik will regularly conduct internal training and participate in architectural discussions, acting as a mentor for junior colleagues. The team will benefit from his improved on-call procedures and playbooks: MTTR will decrease, postmortem quality will improve, and business trust will be enhanced. As a result, Berik will become a key resource in the team, capable of taking on a wider range of technical and organizational tasks.

Conclusion

Berik already possesses important practical skills: a pragmatic approach to CI/CD, good recovery practices, and a mature view on monitoring and incident response. This PDP is aimed at converting these strengths into formalized processes and closing the structural gaps identified in the testing. The plan provides clear, measurable steps for each two-month cycle and includes technical training, practical exercises, and mentorship. The company is ready to support Berik with resources, access to test environments, and time to participate in workshops and simulations—investing in his development is an investment in the reliability and security of Fintech products. The successful completion of this plan will lead to an increase in Berik's professional confidence, a reduction in operational risks, and an increase in his contribution to the team; management values him and expects progress.