Ministry of SMEs and Startups

KOSME Korea SMEs and Startups Agency

LIKELION

JOBKOREA

LESSON 47
Deployment Environment Rehearsal

WEEK 10

Introduction to Deployment Environment Rehearsal

❖ What is a Deployment Environment Rehearsal?

- A deployment environment rehearsal is a simulated test of a software application in an environment that mimics the live production environment. The goal is to verify that everything will work smoothly during the actual deployment.

❖ Why is it important?

- Ensures that potential issues, such as configuration mismatches or performance bottlenecks, are identified and addressed before the real deployment.

❖ Key Benefits of Rehearsals:

- Minimizes downtime during actual deployment.
- Reduces the risk of production failures.
- Provides confidence to the deployment team.

Types of Deployment Environments

- **Development Environment**
  - Used by developers to write and test code. It's typically unstable and may not represent the final version of the application.
- **Staging Environment**
  - A replica of the production environment used for final testing before deployment. It should resemble the production environment as closely as possible.
- **Production Environment**
  - The live environment where the application is actually used by end users. It must be stable and highly available.

Key Considerations for Deployment Rehearsal

❖ Configuration Management

- ➢ Ensuring that all configurations in the rehearsal environment match those of the production environment. Any discrepancies could lead to issues during the real deployment.

❖ Database Migration

- ➢ Ensure that database changes (schema updates, migrations) are tested thoroughly in a controlled environment before the actual deployment.

❖ Environment Variables

- ➢ Test environment variables to ensure they are correctly configured and accessible in the rehearsal environment.

Tools for Deployment Environment Rehearsal

- **Docker**
  - Used to containerize the application for a consistent environment across development, staging, and production.
- **Kubernetes**
  - Manages containerized applications, ensuring smooth orchestration and scaling during deployment rehearsals.
- **CI/CD Pipelines**
  - Automates the deployment process, making it easier to test and deploy in different environments without manual intervention.

Creating a Rehearsal Plan

❖ Define the Objectives

- ➢ What is the rehearsal trying to achieve? For example, ensure code stability, test migration processes, or simulate production traffic.

❖ Identify Key Risks

- ➢ What are the potential risks during deployment? For example, unexpected downtime, database failures, or performance issues.

❖ Allocate Resources

- ➢ Ensure the team has the necessary resources (hardware, software, human resources) to simulate the production environment effectively.

Common Issues During Deployment Rehearsal

❖ Configuration Mismatches

- ➢ Different configurations in rehearsal and production environments can lead to bugs or crashes in the live environment.

❖ Performance Bottlenecks

- ➢ Simulated production loads may reveal performance bottlenecks that were not noticeable in development.

❖ Security Flaws

- ➢ Missing security patches or misconfigured access controls can lead to vulnerabilities.

Example: Rehearsing Database Migration

- **Scenario:**
  - The rehearsal involves testing a database migration that includes new schema updates.
- **Steps:**
  - Backup current production data.
  - Apply migration scripts in the rehearsal environment.
  - Verify that the migration is successful and no data is lost.
- **Expected Outcome:**
  - The migration should complete without errors, and all application features should continue to work as expected.

Monitoring and Logging During Rehearsal

❖ Importance of Monitoring

- ➢ Helps detect issues early by tracking metrics such as CPU usage, memory usage, and response times during the rehearsal.

❖ Log Collection

- ➢ Collect logs from the application, database, and infrastructure to ensure all systems are functioning as expected.

❖ Real-Time Alerts

- ➢ Set up alerts to notify the team of critical failures or anomalies during rehearsal.

Review and Final Steps

❖ Evaluate Results

- ➢ After the rehearsal, evaluate the results against the defined objectives. Did the environment function as expected?

❖ Adjust and Retest

- ➢ If issues were found, make necessary adjustments and run another rehearsal if required.

❖ Deploy to Production

- ➢ Once the rehearsal is successful, the deployment can proceed with confidence.

Rehearsal Case Study: E-commerce Application

❖ Background

- ➢ An e-commerce company is preparing for a major update to its platform, including new features and a database migration.

❖ Rehearsal Goals

- ➢ Ensure that the new features function as expected.
- ➢ Test the database migration with real data loads.
- ➢ Verify the scalability of the application under heavy traffic.

❖ Steps Taken

- ➢ A full replica of the production environment was created.
- ➢ Load testing was performed with simulated traffic.
- ➢ The migration was applied to a copy of the live database.

Load Testing During Rehearsal

- **Why Load Testing is Crucial**
  - Load testing helps identify how the application behaves under high traffic or usage and ensures that the system can scale effectively during deployment.
- **Rehearsal Scenario:**
  - Simulate thousands of users accessing the site concurrently to test how it performs.
- **Expected Results:**
  - The application should maintain a response time of under 2 seconds, even under high traffic.

CI/CD in Rehearsals

- **What is CI/CD?**
  - CI/CD refers to the practice of continuously integrating code changes and automatically deploying them to various environments.
- **How CI/CD Helps in Rehearsals:**
  - CI/CD pipelines ensure that the latest changes are automatically tested and deployed in the rehearsal environment, ensuring that no errors are introduced.
- **Benefits of CI/CD in Deployment Rehearsals:**
  - Faster feedback loops.
  - Reduced human error.
  - Consistency across environments.

Stress Testing: Pushing Beyond Normal Limits

❖ What is Stress Testing?

- ➢ Stress testing involves pushing the system beyond its limits to see how it behaves under extreme conditions.

❖ Rehearsal Scenario:

- ➢ Simulate an extreme load (e.g., 10x more traffic than expected) to test how the system handles unexpected spikes.

❖ Expected Outcome:

- ➢ The system should fail gracefully, meaning it should either recover without crashing or provide a meaningful error message to the user.

Automated Testing for Deployment Rehearsals

❖ What is Automated Testing?

- ➢ Automated testing involves using scripts and tools to automatically test code, functionality, and performance before deployment.

❖ Tools Used for Automated Testing:

- ➢ Selenium: Automates browser interactions to test UI/UX.
- ➢ JUnit: Runs unit tests to ensure individual components work as expected.
- ➢ LoadRunner: Simulates multiple users to test performance under load.

❖ Benefits:

- ➢ Saves time by running tests quickly and consistently.
- ➢ Helps catch bugs early in the process.

Importance of Security Testing During Rehearsal

- **Security Risks During Deployment**
  - Security vulnerabilities can cause data breaches or make the application susceptible to attacks.
- **Security Testing Scenarios:**
  - Penetration testing: Simulating an attack on the application to find vulnerabilities.
  - Dependency scanning: Ensuring that no third-party libraries have known vulnerabilities.
- **Outcome:**
  - All security flaws should be identified and patched before deployment.

Backup / Rollback Strategy for Deployment Rehearsal

❖ Why is a Backup Strategy Important?

- ➢ A backup strategy ensures that data and configurations can be restored if something goes wrong during deployment.

❖ Rollback Scenarios:

- ➢ A major bug is discovered post-deployment.
- ➢ Database migration fails and causes data corruption.

❖ Steps to Implement:

- ➢ Regular backups of databases and configurations should be taken before any changes are applied.

Monitoring Tools for Deployment Rehearsal

❖ Why Monitoring is Critical

- ➢ Continuous monitoring helps detect issues early, ensuring that the deployment rehearsal is on track.

❖ Common Monitoring Tools:

- ➢ Prometheus: Collects metrics on application performance.
- ➢ Grafana: Visualizes monitoring data to spot performance bottlenecks.
- ➢ New Relic: Provides real-time insights into system performance.

❖ How Monitoring Helps:

- ➢ Allows the team to act immediately on issues such as slow response times or high memory usage.

Monitoring Tools for Deployment Rehearsal

❖ Why Monitoring is Critical

- ➢ Continuous monitoring helps detect issues early, ensuring that the deployment rehearsal is on track.

❖ Common Monitoring Tools:

- ➢ Prometheus: Collects metrics on application performance.
- ➢ Grafana: Visualizes monitoring data to spot performance bottlenecks.
- ➢ New Relic: Provides real-time insights into system performance.

❖ How Monitoring Helps:

- ➢ Allows the team to act immediately on issues such as slow response times or high memory usage.