PW SKILLS | DevOps and Cloud Computing

Advanced Continuous Deployment

Objective

- Understand different deployment strategies: Blue/Green, Canary, and Rolling.
- Implement automated testing and monitoring in CD pipelines.
- Learn deployment automation and orchestration techniques.
- Apply CD best practices to improve efficiency and reliability
- Understand release management processes and approval workflows.

PW SKILLS

Explain deployment strategies

Deployment strategies

1. **Blue/Green Deployment**

- This strategy involves maintaining two identical environments: one active (blue) and one standby (green). The new version is deployed to the green environment, tested, and then traffic is switched from blue to green.
- This approach ensures zero-downtime deployments and allows for easy rollbacks if issues arise.

Use Case: Ideal for applications where downtime must be minimized and rollbacks are crucial.

Deployment strategies

2. Canary Deployment

- Involves deploying the new version to a small subset of users (the canary group) before rolling it out to the entire user base.
- This approach helps identify issues early and minimizes the impact on users.

Use Case: Suitable for frequent updates and testing, especially in environments where user feedback is valuable.

Deployment strategies

3. Rolling Deployment

- Gradually replaces instances of the old version with the new version in a controlled sequence.
- This strategy allows for zero-downtime deployments by ensuring that some instances are always available.

Use Case: Ideal for applications requiring continuous availability with minimal risk of downtime.

PW SKILLS

Comparing the benefits and trade-offs of each approach

Benefits & Trade-offs

1. **Blue/Green Deployment:**

Benefits:

- Zero-Downtime Deployments
- Easy Rollbacks
- Performance Testing
- Disaster Recovery

Trade-Offs:

- Higher Costs
- Complexity
- Database Challenges

Benefits & Trade-offs

2. Canary Deployment

Benefits:

- Gradual Rollout
- Real-World Testing
- Controlled Risk

Trade-Offs:

- User Segmentation
- Complexity
- Resource Intensive

Benefits & Trade-offs

3. Rolling Deployment

Benefits:

- Continuous Availability
- Efficient Resource Use
- Simplified Rollbacks

Trade-Offs:

- Risk of Partial Downtime
- Complexity in Rollbacks
- Testing Challenges

PW SKILLS

The importance of automated testing in CD pipelines

Importance of automated testing

Automated testing is crucial in Continuous Delivery (CD) pipelines for several reasons:

Benefits:

- Early Bug Detection
- Faster Feedback Loops
- Improved Quality
- Efficiency and Speed

Importance of automated testing

Types of Automated Tests:

- Unit Tests: Validate individual functions to ensure they work as expected.
- Integration Tests: Verify interactions between services to ensure seamless functionality.
- Smoke Tests: Confirm basic functionality in production environments to ensure deployments are successful.

Role in CD Pipelines:

- Continuous Quality Assurance
- Supports Agile Practices

PW SKILLS

Introducing monitoring tools

Monitoring tools

1. **Prometheus:**

An open-source monitoring system that collects metrics from systems and applications. It provides a robust alerting system based on user-defined rules.

Use Case: Ideal for tracking performance metrics, resource usage, and system health in real-time.

2. **Grafana:**

An open-source visualization tool that integrates with various data sources to create customizable dashboards. It is often used with Prometheus for real-time monitoring.

Use Case: Suitable for visualizing metrics and creating interactive dashboards to monitor CI/CD pipeline performance.

Monitoring tools

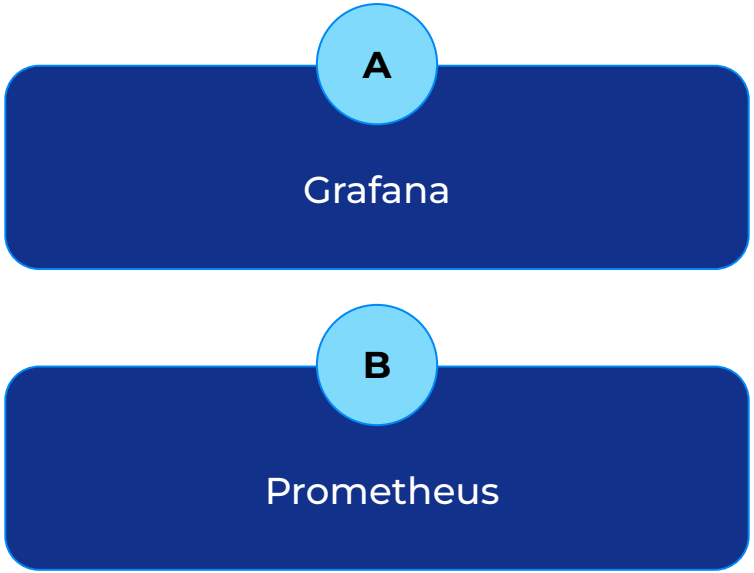
3. ELK Stack (Elasticsearch, Logstash, Kibana)
A log monitoring and analysis toolset.

- Elasticsearch: Stores and indexes log data.
- Logstash: Collects, processes, and forwards logs.
- Kibana: Provides a user interface for visualizing and analyzing log data.

Use Case: Ideal for log aggregation, analysis, and visualization to identify issues and trends in application logs.

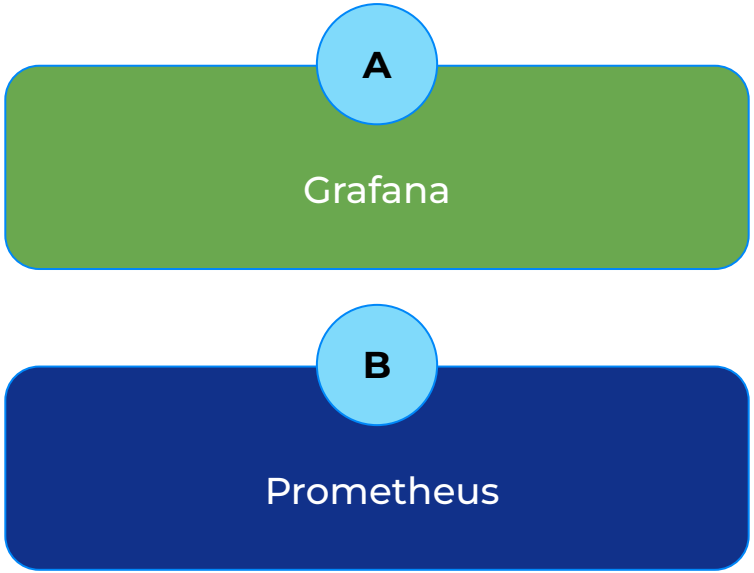
Pop Quiz

Q. Which tool is primarily used for visualizing monitoring data?



Pop Quiz

Q. Which tool is primarily used for visualizing monitoring data?



Take A 5-Minute Break!

- Stretch and relax
- Hydrate
- Clear your mind
- Be back in 5 minutes

PW SKILLS

Explaining the role of orchestration in CD pipelines

Role of Orchestration

Orchestration in Continuous Deployment (CD) pipelines plays a crucial role by coordinating automated tasks across multiple systems to ensure that different components work together seamlessly. Here's a concise overview of its role:

- Coordination and Governance
- Dynamic Management
- Automation and Efficiency
- Scalability and Resilience

Pop Quiz

Q. What is the primary role of orchestration in CD pipelines?

A

To manage and coordinate the steps in a CI/CD pipeline

B

To automate testing and code integration

Pop Quiz

Q. What is the primary role of orchestration in CD pipelines?

A

To manage and coordinate the steps in a CI/CD pipeline

B

To automate testing and code integration

PW SKILLS

Introduce orchestration tools

Orchestration tools

Kubernetes
Overview: Kubernetes is an open-source container orchestration tool that automates the deployment, scaling, and management of containerized applications. It was originally developed by Google and is now maintained by the Cloud Native Computing Foundation.

Terraform
Overview: Terraform is an Infrastructure as Code (IaC) tool that allows users to define and manage cloud and on-premises infrastructure using human-readable configuration files. It supports a wide range of cloud providers and services.

Spinnaker
Overview: Spinnaker is a multi-cloud continuous delivery platform that helps manage the deployment of software releases across different environments. It supports various cloud providers and provides features like automated canary releases and rollbacks.

PW SKILLS

Demonstrating setting up an automated deployment pipeline using Kubernetes

Let’s do it

Components Needed:

- Kubernetes Cluster
- CI/CD Tool
- Containerization

Steps to Set Up the Pipeline:

1. Configure CI/CD Pipeline
2. Build and Push Container Image
3. Deploy to Kubernetes
4. Implement Rollback Mechanism
5. Monitor and Optimize

Let’s do it

Example Jenkinsfile for Kubernetes Deployment:

```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                sh 'docker build -t myapp .'
                sh 'docker tag myapp:latest myapp:latest'
                sh 'docker push myapp:latest'
            }
        }
        stage('Deploy') {
            steps {
                sh 'kubectl apply -f deployment.yaml'
            }
        }
    }
}
```

PW SKILLS

The best practices for optimizing CD pipelines

Let’s discuss

1. **Keep Deployments Small and Frequent**

- Benefits: Small deployments reduce the risk of failures and make it easier to identify and fix issues quickly.
- Implementation: Use incremental builds and deploy only the changed components. This approach is facilitated by microservices architecture, where each service can be updated independently

Let’s discuss

2. Automate Rollback Mechanisms for Failed Deployments

- Benefits: Automated rollbacks ensure that if a deployment fails, the system can quickly revert to a stable state, minimizing downtime.
- Implementation: Use tools like Kubernetes to automate rollbacks. Implement health checks to detect failures and trigger rollbacks automatically.

3. Ensure Security and Compliance Checks Are Integrated into CD Workflows

- Benefits: Integrating security checks early in the pipeline helps identify vulnerabilities before they reach production, ensuring compliance with regulatory standards.
- Implementation: Use tools like Snyk or OWASP ZAP to scan for vulnerabilities. Implement compliance checks to ensure adherence to industry standards and regulations.

PW SKILLS

Explaining the importance of approvals in enterprise CD workflows

Let’s see

Approvals in enterprise Continuous Deployment (CD) workflows are crucial for ensuring that deployments meet organizational standards and requirements. Here's a concise explanation of their importance:

- Compliance and Governance
- Risk Mitigation
- Efficiency and Transparency
- Quality Assurance

PW SKILLS

Discussing automated vs. manual approvals and governance in CD

Let’s discuss

1. **Automated Approvals**

Automated approvals use predefined checks and tests to validate deployments before moving to the next stage.

- This approach is typical in continuous deployment, where code changes are automatically deployed to production if they pass all automated tests.

Let’s discuss

2. Manual Approvals
Manual approvals involve human intervention to review and approve deployments before they proceed to the next stage.

- This is common in continuous delivery, where deployments to production require a manual approval step.

3. Governance in CD
Governance ensures that deployments adhere to organizational policies and regulatory standards.

- It involves setting up guardrails and automated checks to enforce compliance without necessarily requiring manual approvals.

PW SKILLS

Demonstrating setting up approval workflows in GitLab CI/CD or Jenkins

Let’s do it

GitLab CI/CD Approval Workflow

1. **Configure Protected Environments:**
   - Go to Settings > CI/CD > Protected environments.
   - Create or select an environment (e.g., production) that requires approval.

2. **Add Approval Rules:**
   - For each environment, add approval rules specifying who can approve deployments.
   - Set the required number of approvals needed before a deployment can proceed.

3. **Implement in .gitlab-ci.yml:**
   - Ensure your deployment job is configured to deploy to the protected environment.

Let’s do it

Example:

```
stages:
  - deploy

production:
  stage: deploy
  script:
    - 'echo "Deploying to ${CI_ENVIRONMENT_NAME}"'
  environment:
    name: ${CI_JOB_NAME}
```

Let’s do it

Jenkins Approval Workflow

1. **Add Manual Approval Stage:**

- Use the ‘input’ directive in your Jenkinsfile to pause the pipeline for approval.

Example:

```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                // Build steps
            }
        }
        stage('Approval') {
            steps {
                timeout(time: 15, unit: "MINUTES") {
                    input message: 'Do you want to approve the deployment?', ok: 'Yes'
                }
            }
        }
        stage('Deploy') {
            steps {
                // Deployment steps
            }
        }
    }
}
```

Let’s do it

2. Configure Notifications:

- Optionally, set up email or Telegram notifications to alert approvers.

Example for Telegram:

```
environment {
    TOKEN = credentials('telegram-api')
    CHAT_ID = credentials('telegram_chatid')
}

// ...

post {
    success {
        sh "curl -X POST -H 'Content-Type: application/json' -d
'{\"chat_id\": \"${CHAT_ID}\", \"text\": \"Deploy succeeded!\",
\"disable_notification\": false}'
\"https://api.telegram.org/bot${TOKEN}/sendMessage\""
    }
}
```

PW SKILLS

Time for case study!

Important

- Complete the post-class assessment
- Complete assignments (if any)
- Practice the concepts and techniques taught in this sessio
- Review your lecture notes
- Note down questions and queries regarding this session ar
  consult the teaching assistants

Thanks !