Masterclass Certificate in AI-Driven Release Management

Infrastructure as Code and Configuration Management

Agile Infrastructure: Agile Infrastructure is a approach to managing and provisioning technology resources that emphasizes flexibility, speed, and efficiency. It is often associated with the Agile methodology in software development, which values responsiveness to changing requirements and close collaboration between development and operations teams. Agile Infrastructure typically involves the use of automation tools and Infrastructure as Code (IaC) practices to enable rapid deployment and configuration of infrastructure resources.

Automated Testing: Automated testing is the use of software tools to run tests on software applications, rather than manually executing test cases. Automated testing can help to improve the speed, accuracy, and consistency of testing, and is often used in conjunction with Agile and DevOps practices to enable frequent and reliable releases. Automated tests can be written in a variety of programming languages and frameworks, and can be integrated into continuous integration and delivery pipelines to provide fast feedback on the quality of code changes.

Blue-Green Deployment: Blue-Green Deployment is a release management strategy that involves running two identical production environments, called Blue and Green, in parallel. At any given time, one of these environments is live and serving traffic, while the other is idle. When a new release is ready, it is deployed to the idle environment and thoroughly tested. Once the new release has been validated, traffic is gradually shifted from the live environment to the new one, allowing for zero-downtime deployments and easy rollbacks in case of issues.

Canary Release: Canary Release is a release management strategy that involves deploying a new release to a small subset of users or servers, and gradually increasing the scale of the deployment over time. This allows teams to test the new release in a controlled manner and gather feedback from real users before rolling it out to the entire user base. Canary Releases can help to reduce the risk of downtime or issues associated with new releases, and can provide valuable insights into the performance and stability of the new code.

Configuration Drift: Configuration Drift is the gradual divergence of the actual configuration of a system from its intended or documented configuration. This can occur due to manual changes, automated processes, or other factors, and can lead to inconsistencies, errors, and security vulnerabilities. Configuration Management tools and practices, such as Infrastructure as Code, can help to prevent Configuration Drift by ensuring that configurations are consistently applied and tracked across all systems and environments.

Continuous Integration (CI): Continuous Integration is a software development practice in which code changes are frequently merged into a shared repository and automatically built, tested, and deployed. CI helps to ensure that code changes are properly integrated, tested, and validated, and can reduce the risk of integration issues and conflicts. Continuous Integration is often used in conjunction with Continuous Delivery and Continuous Deployment to enable frequent and reliable releases.

Continuous Delivery (CD): Continuous Delivery is a software development practice in which code changes are automatically built, tested, and deployed to a production environment, ready for release at any time. CD helps to ensure that code changes are properly validated and deployed in a consistent and reliable manner, and can reduce the time and effort required for releases. Continuous Delivery is often used in conjunction with Continuous Integration and Continuous Deployment to enable frequent and reliable releases.

Continuous Deployment (CD): Continuous Deployment is a software development practice in which code changes are automatically deployed to a production environment as soon as they are tested and validated. CD helps to reduce the time and effort required for releases, and can enable teams to deliver value to users more quickly and frequently. Continuous Deployment is often used in conjunction with Continuous Integration and Continuous Delivery to enable frequent and reliable releases.

DevOps: DevOps is a culture and practice that emphasizes collaboration and communication between development and operations teams, with the goal of improving the speed, quality, and reliability of software development and delivery. DevOps often involves the use of Agile, Lean, and Systems Thinking principles, as well as automation tools and Infrastructure as Code practices, to enable rapid and reliable deployment and operation of software systems.

Golden Image: A Golden Image is a preconfigured and tested image of a system or environment that can be used as a baseline for deployment and configuration. Golden Images can help to ensure consistency, reliability, and security across multiple systems and environments, and can reduce the time and effort required for deployment and configuration. Golden Images are often used in conjunction with Configuration Management tools and practices, such as Infrastructure as Code.

Immutable Infrastructure: Immutable Infrastructure is a practice in which infrastructure resources, such as servers or containers, are never modified after they are deployed. Instead, any changes or updates are applied by deploying new resources and discarding the old ones. Immutable Infrastructure can help to improve the reliability, security, and scalability of infrastructure resources, and can reduce the risk of configuration drift and errors.

Infrastructure as Code (IaC): Infrastructure as Code is a practice in which infrastructure resources, such as servers, networks, and storage, are defined and managed using code-based configuration files, rather than manual processes or ad-hoc tools. IaC enables infrastructure to be treated as software, with all the benefits

of version control, testing, and automation. IaC can help to improve the speed, consistency, and reliability of infrastructure provisioning and configuration, and is often used in conjunction with DevOps and Agile practices.

Monitoring: Monitoring is the practice of continuously observing and analyzing the performance, availability, and other metrics of a system or application, with the goal of identifying and resolving issues before they impact users. Monitoring can help to ensure that systems are running smoothly and efficiently, and can provide valuable insights into the health and behavior of complex systems. Monitoring tools and practices can be integrated into continuous integration and delivery pipelines to provide fast feedback and enable rapid response to issues.

Pipelines: Pipelines are automated workflows that define the steps and processes involved in building, testing, and deploying software applications. Pipelines can help to improve the speed, consistency, and reliability of software development and delivery, and can reduce the time and effort required for releases. Continuous Integration, Continuous Delivery, and Continuous Deployment are often implemented using pipelines.

Rolling Deployment: Rolling Deployment is a release management strategy that involves deploying new code to a small subset of servers or containers, and gradually increasing the scale of the deployment over time. This allows teams to test the new code in a controlled manner and minimize the risk of downtime or issues associated with new releases. Rolling Deployments can help to reduce the risk of failures and outages, and can provide a smooth and seamless experience for users.

Serverless Computing: Serverless Computing is a cloud computing model in which the underlying infrastructure, such as servers and storage, is managed by the cloud provider, and users focus on building and deploying applications and services. Serverless Computing can help to reduce the operational overhead and costs associated with infrastructure management, and can enable teams to focus on delivering value to users. Serverless Computing is often used in conjunction with Infrastructure as Code and DevOps practices.

Test-Driven Development (TDD): Test-Driven Development is a software development practice in which tests are written before the code, and the code is written to pass the tests. TDD helps to ensure that code is properly tested, verified, and validated, and can reduce the risk of bugs and errors. TDD is often used in conjunction with Continuous Integration and Continuous Delivery to enable frequent and reliable releases.

Version Control: Version Control is a practice in which changes to code or configuration files are tracked and managed using a version control system, such as Git or Subversion. Version Control enables teams to collaborate and coordinate on code changes, and can help to ensure that code is properly tested, verified, and validated. Version Control is often used in conjunction with Continuous Integration and Continuous Delivery to enable frequent and reliable releases.

Zero-Downtime Deployment: Zero-Downtime Deployment is a release management strategy that aims

to minimize or eliminate downtime associated with new releases. Zero-Downtime Deployments can be achieved using various techniques, such as Blue-Green Deployment, Canary Release, or Rolling Deployment, and can help to ensure that users have a smooth and seamless experience during releases. Zero-Downtime Deployments can reduce the risk of failures and outages, and can improve the reliability and availability of systems and applications.