4-1-1: Examining IaC and CaC

After completing this episode, you should be able to:

- Identify and explain using code for deployment and configuration in the cloud, given a scenario.

Description: In this episode, the learner will examine using code to provision and configure cloud-based resources. We will explore Infrastructure-as-Code or IaC, Configuration-as-Code or CaC, drift detection, repeatability, testing, and more.

- Describe Infrastructure as Code (IaC)
    - Provisioning computing environments through text-based files rather than physical hardware configuration or interactive configuration tools
    - Replacing manual processes to provision computing resources
    - Benefits
        - ♠ Standardization
        - ♠ Speed
        - ♠ Consistency
        - ♠ Repeatability
- Describe Configuration as Code (CaC)
    - Using text-based definition files to manage the configuration of systems
    - Configuration is measured against a desired state, such as a baseline
    - Helps to ensure configurations are applied consistently across infrastructure
    - Benefits
        - ♠ Standardization
        - ♠ Speed
        - ♠ Consistency
        - ♠ Repeatability
- Describe the significance of repeatability:
    - The ability to consistently recreate the same immutable environment multiple times without manual intervention, crucial for minimizing variations between development and production environments
- Describe the importance of versioning in code-based deployments
    - Versioning
        - ♠ Managing changes to configuration scripts and infrastructure setups through version control systems
    - Allows for the tracking of modifications and rollback, if necessary
    - Examples - Git, Bitbucket (Atlassian), AWS CodeCommit, Beanstalk
- Describe configuration drift and drift detection:
    - Configuration drift
        - ♠ A situation in which a system, component, or element's configuration deviates from an established configuration baseline
    - Drift detection
        - ♠ Identifies and reports changes from the defined configuration baseline
        - ♠ Helps in maintaining control over infrastructure and configuration states.
    - Configuration management software - Ansible, SaltStack, Chef, Puppet, Terraform
- Describe the importance of testing
    - Helps to ensure the environment meets the required specifications before deployment
    - Using IaC and CaC can automate the creation of the infrastructure and configuration
    - Leveraging testing automation to reduce manual intervention and optimize the deployments
- Describe the importance of accurate and timely documentation

    - Keeping detailed records of configurations, setups, and changes to support maintenance, compliance, and future modifications
    - Follow change and configuration management policies, procedures, and guidelines

Additional Resources

- Example Scenarios
    - Standardized Development Environments:
        - Scenario: Leverage IaC to automate the creation of development, testing, and production environments that are identical in every aspect.
        - Benefit: Ensures consistency and reduces environment-related issues when moving applications through stages.
    - Real-Time Infrastructure Adjustment:
        - Scenario: Use CaC tools integrated with drift detection mechanisms to automatically adjust configurations in real-time, maintaining the desired state.
        - Benefit: Keeps systems secure and compliant by quickly reverting unauthorized changes.
    - Version-Controlled Infrastructure Updates:
        - Scenario: Apply versioning principles to IaC to manage changes to cloud infrastructure, allowing for controlled updates and quick rollbacks.
        - Benefit: Increases operational reliability by tracking changes and facilitating recovery if updates cause issues.
    - Automated Testing for Infrastructure Deployment:
        - Scenario: Implement automated testing protocols to validate infrastructure and configurations coded in IaC before they go live.
        - Benefit: Reduces deployment failures by catching errors early in the deployment process.