Mitesh Soni

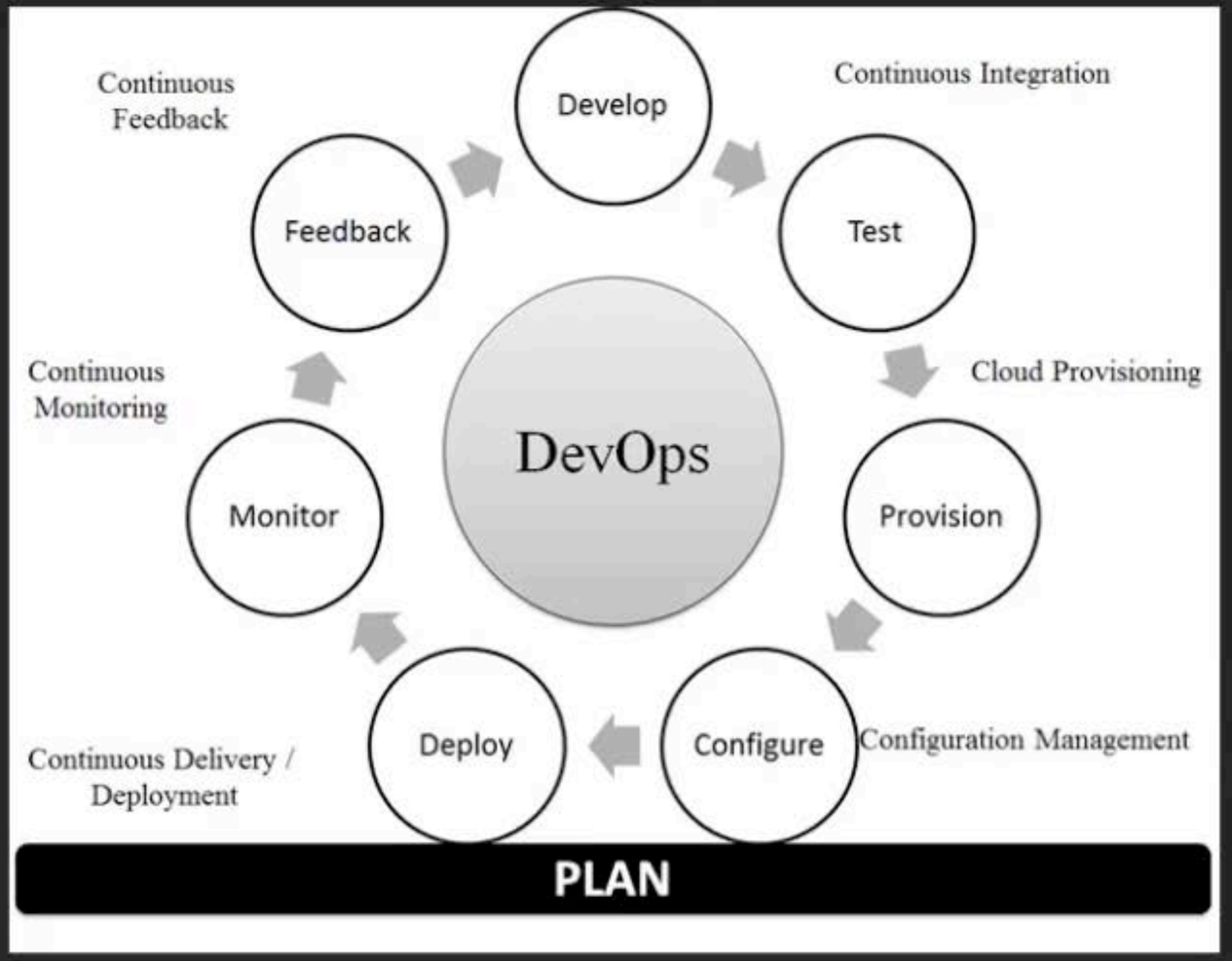
Video 1.3

The DevOps Lifecycle

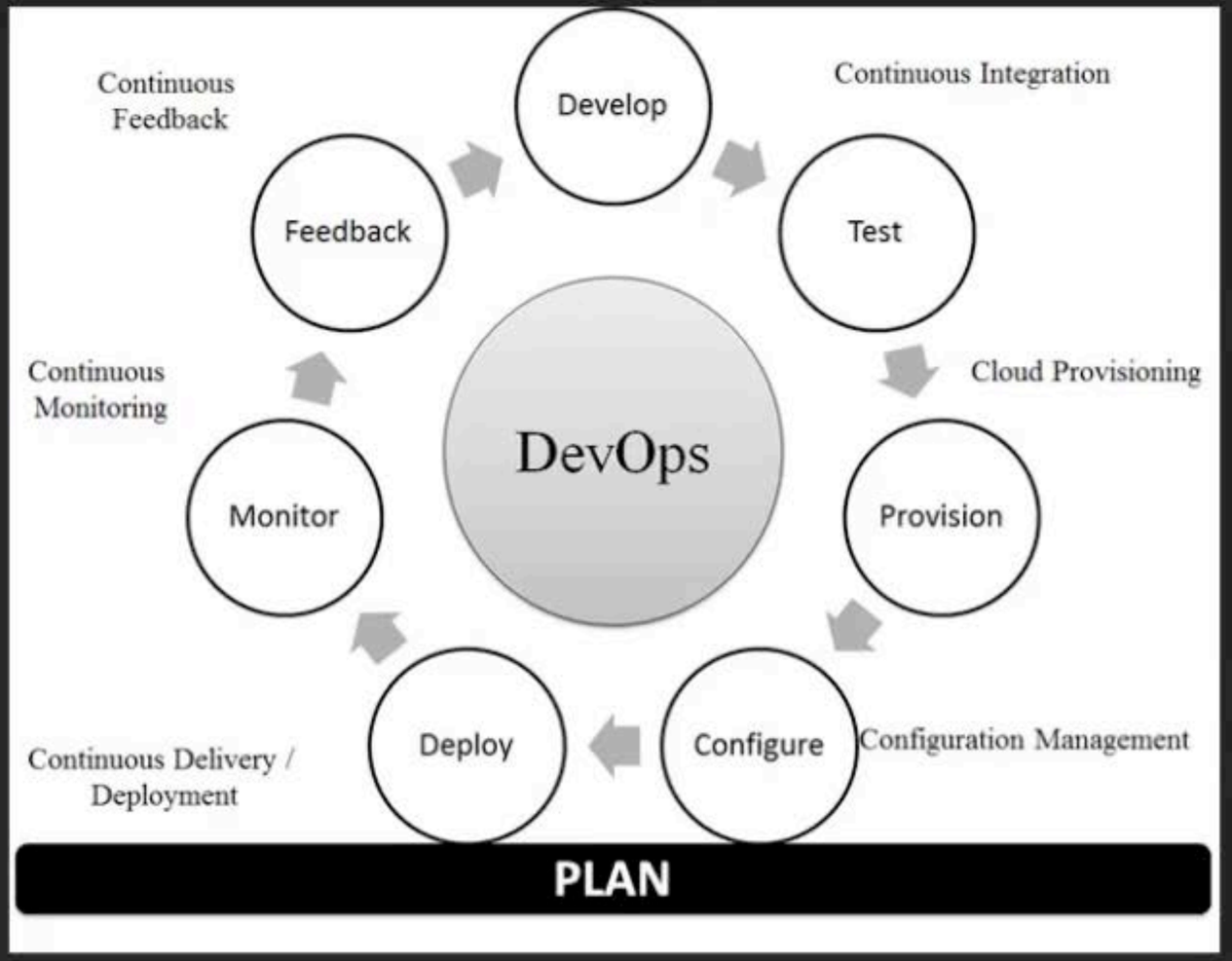
In this Video, we are going to take a look at...

- Building automation
- Automating test case execution
- Executing the scripts based on the requirement

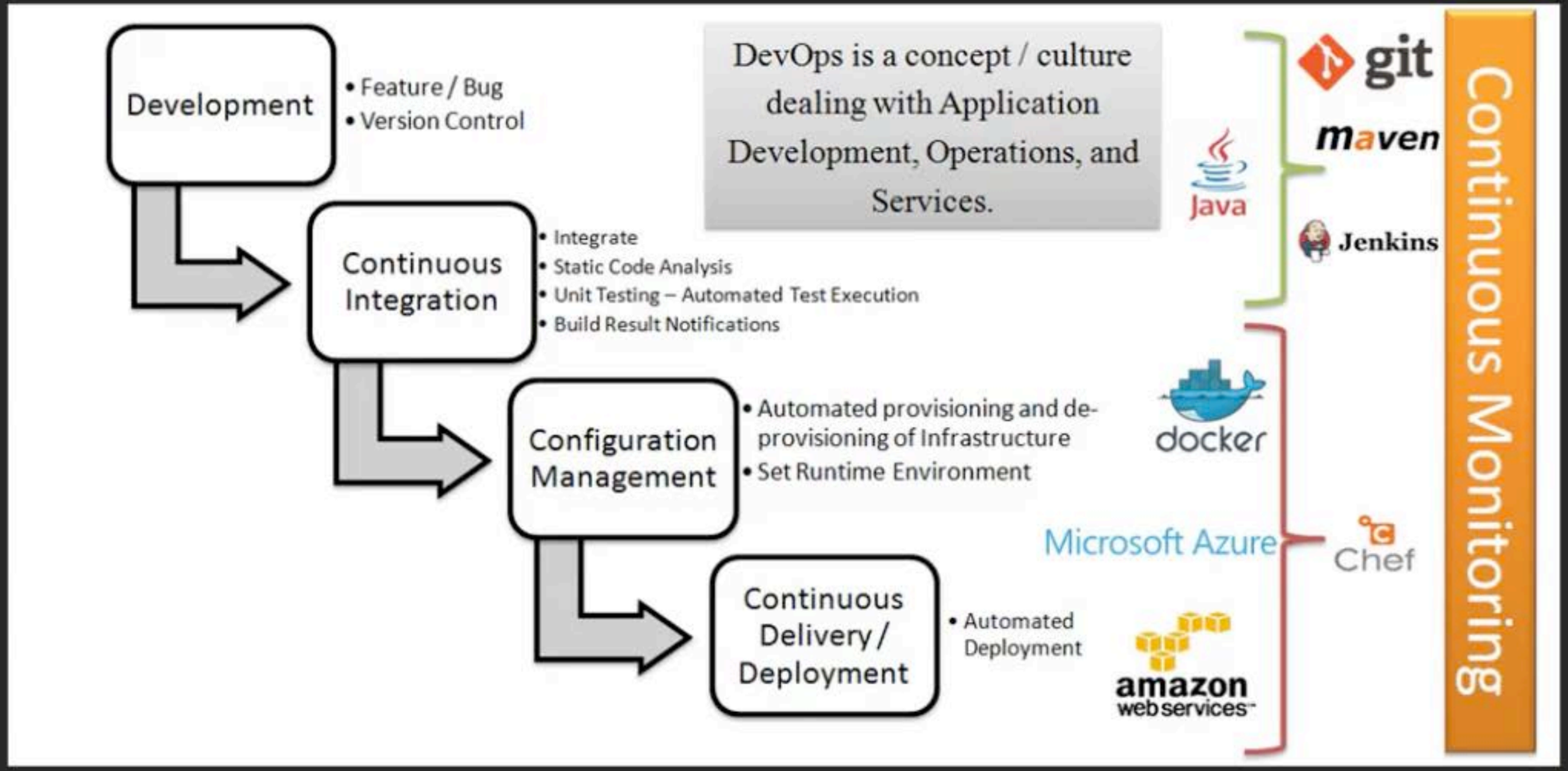
Application Delivery Lifecycle



Application Delivery Lifecycle



Mapping of an Application



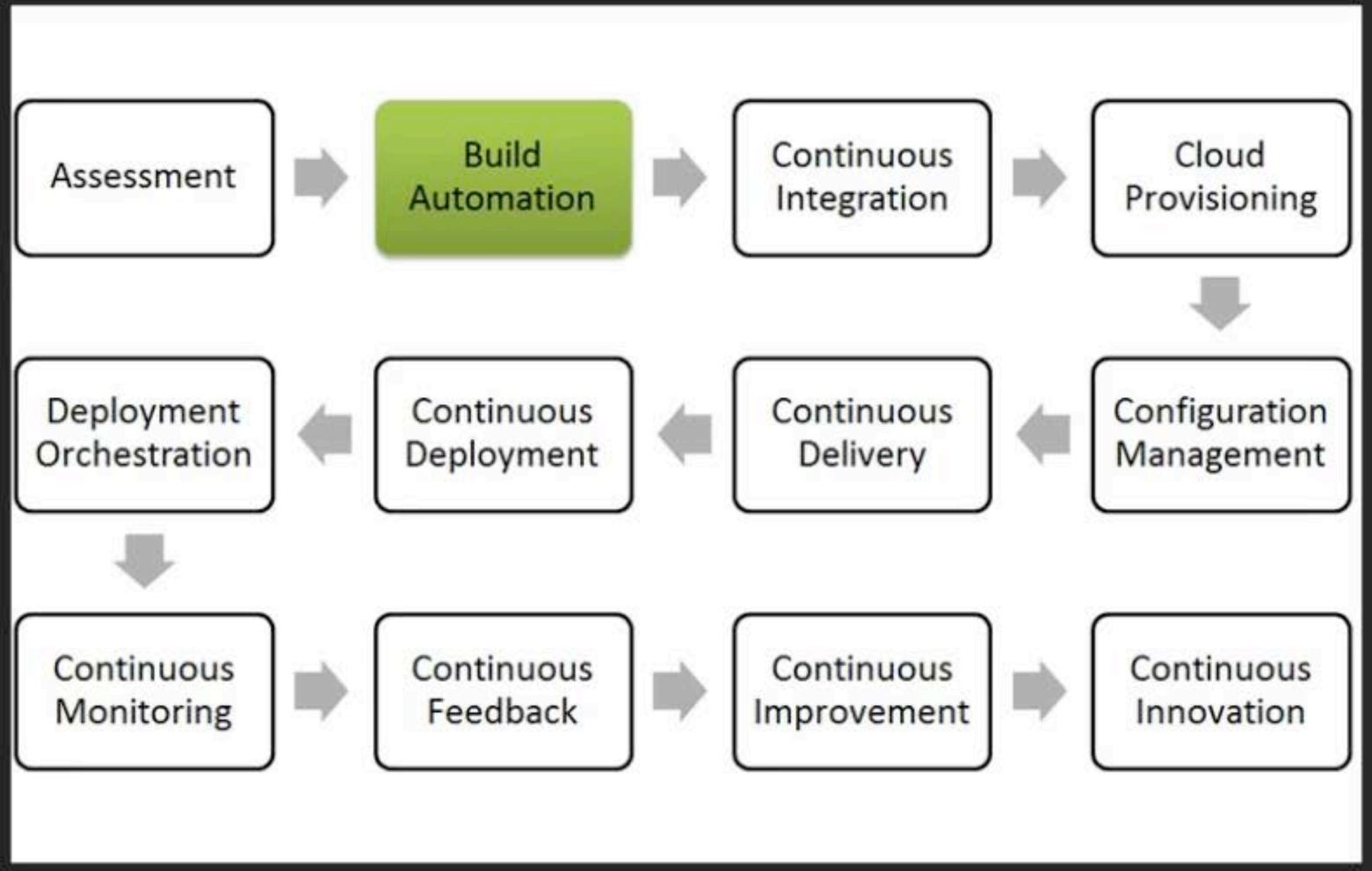
Build Automation

- Compiling source code into class files or binary files
- Providing references to third-party library files
- Providing the path of configuration files
- Packaging class files or binary files into WAR files in the case of Java

Build Automation

- Executing automated test cases
- Deploying WAR files on local or remote machines
- Reducing manual effort in creating the WAR file

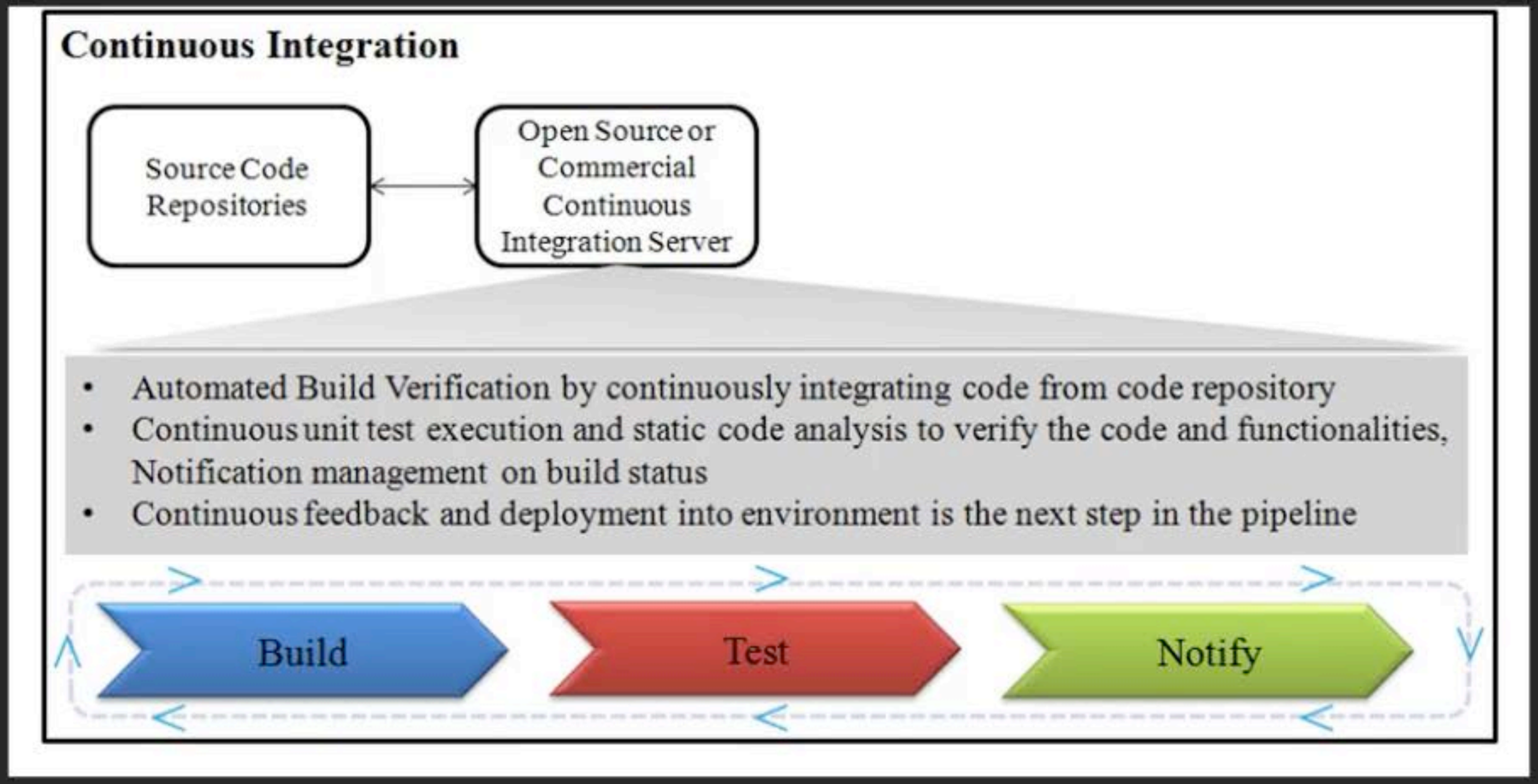
Build Automation Pipeline



Continuous Integration

- Pull mechanism
- Push mechanism

Continuous Integration Pipeline



Needs of CI

- It helps us identify bugs or errors in the code at a very early stage of development
- It is far easier to cure and fix issues
- CI is a development practice that requires developers to integrate code into a shared repository several times a day
- CI is a significant part for the release-management strategy of any organization that wants to develop a DevOps culture

Benefits CI

- Automated integration with pull or push mechanism
- Repeatable process without any manual intervention
- Automated test case execution
- Coding standard verification

Benefits CI

- Execution of scripts based on requirement
- Quick feedback: build status notification to stakeholders via e-mail
- Teams focused on their work and not in the managing processes

Best Practices

- Maintain a code repository such as Git or SVN
- Check-in third-party JAR files, build scripts, other artifacts
- Execute builds fully from the code repository
- Automate the build using Maven or Ant for Java
- Make the build self-testing

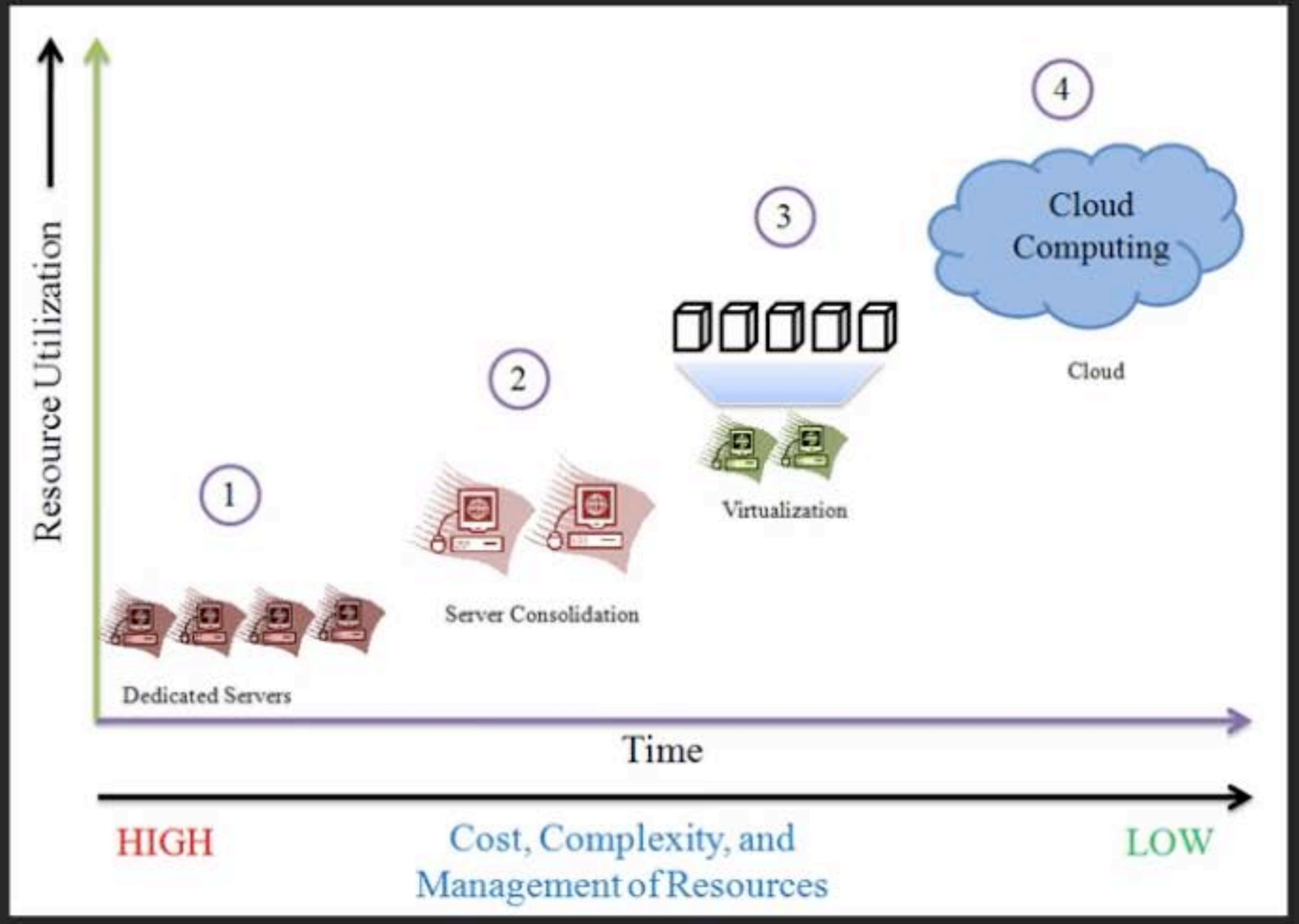
Best Practices

- Commit all changes at least once a day per feature
- Every commit should be built to verify the integrity of changes
- Authentication and authorization
- Use alphanumeric characters for build names and avoid symbols
- Helps to assign build execution to slave instances

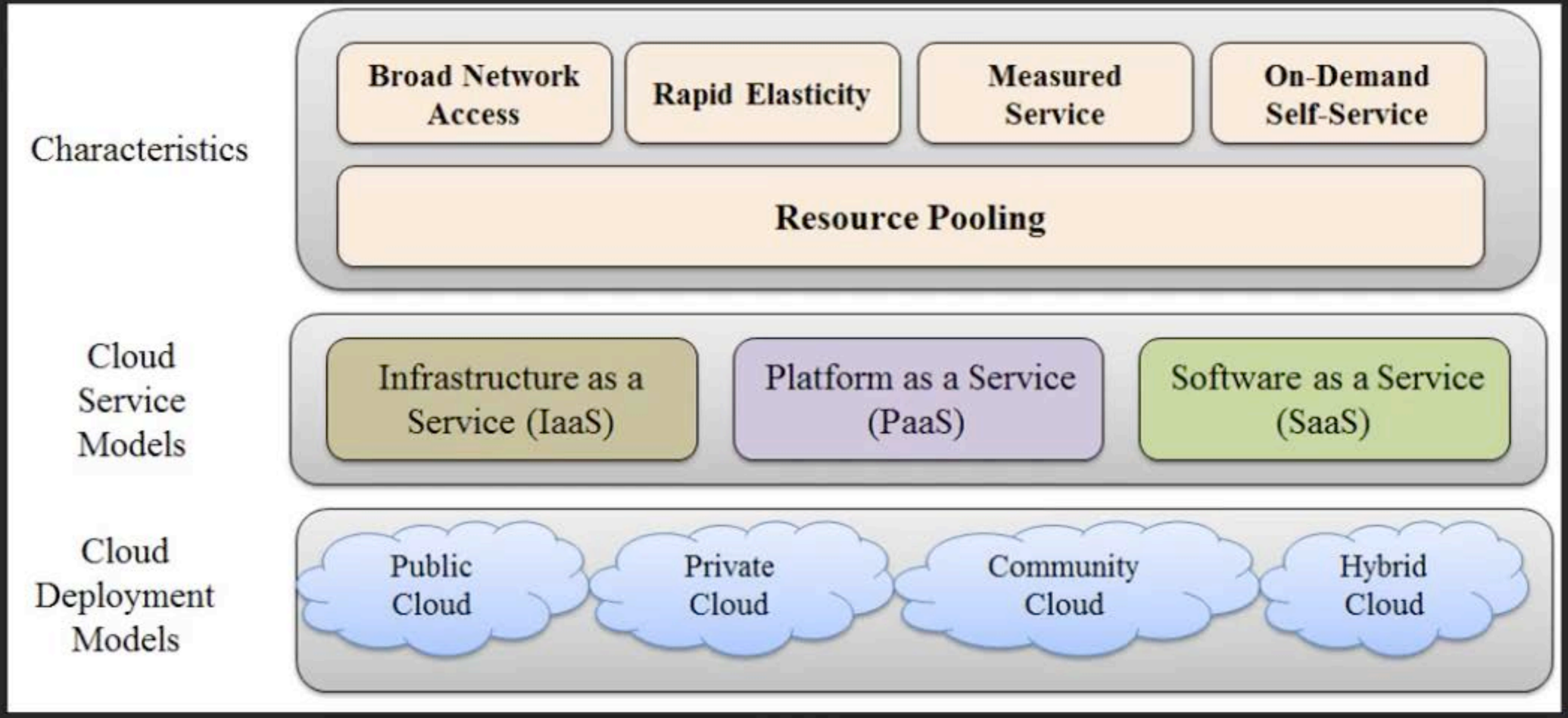
Best Practices

- Backup the home directory of the CI server regularly
- Make sure the CI server has enough free disk space
- Do not schedule multiple jobs to start at the same time, or use a master-slave concept
- Set up an e-mail, SMS, or Twitter notification to specific stakeholders of a project

Cloud Computing



Service Models and Deployment Models



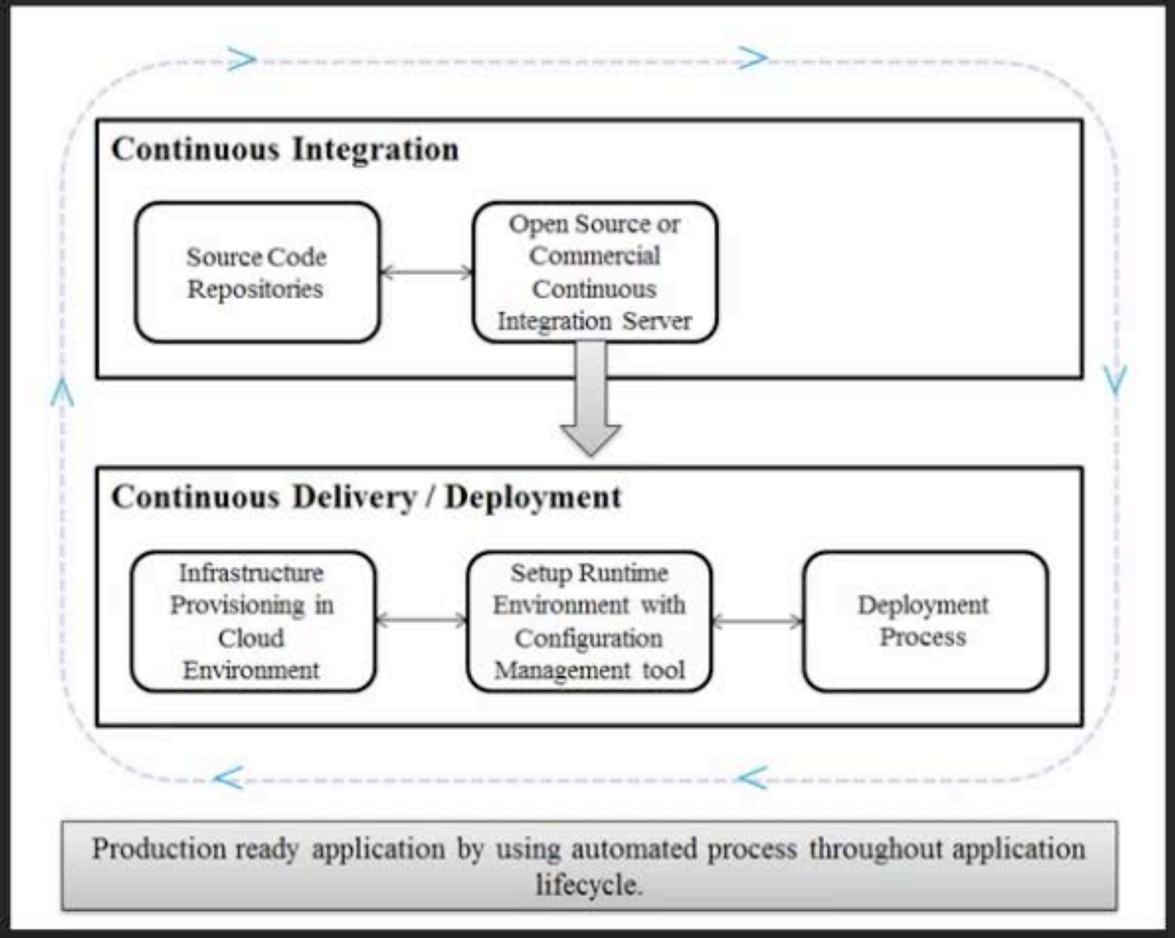
Cloud Deploying Models

- Public Cloud
- Private Cloud
- Community Cloud
- Hybrid Cloud

Configuration Management

- CM is keeping track or versions of details related to the state of specific nodes
- A centralized change can trigger this or nodes can communicate with the CM server
- CM tools make this process efficient when only changed behavior is updated
- The entire installation and modification isn't applied again to the server nodes
- Chef, Puppet, Ansible, and Salt are the popular CM tools

Continuous Delivery/Deployment



Example to Deploy an Application in Microsoft Azure

- The Azure web app configured with specific types of resources
- A storage account to store BACPAC files to create the database

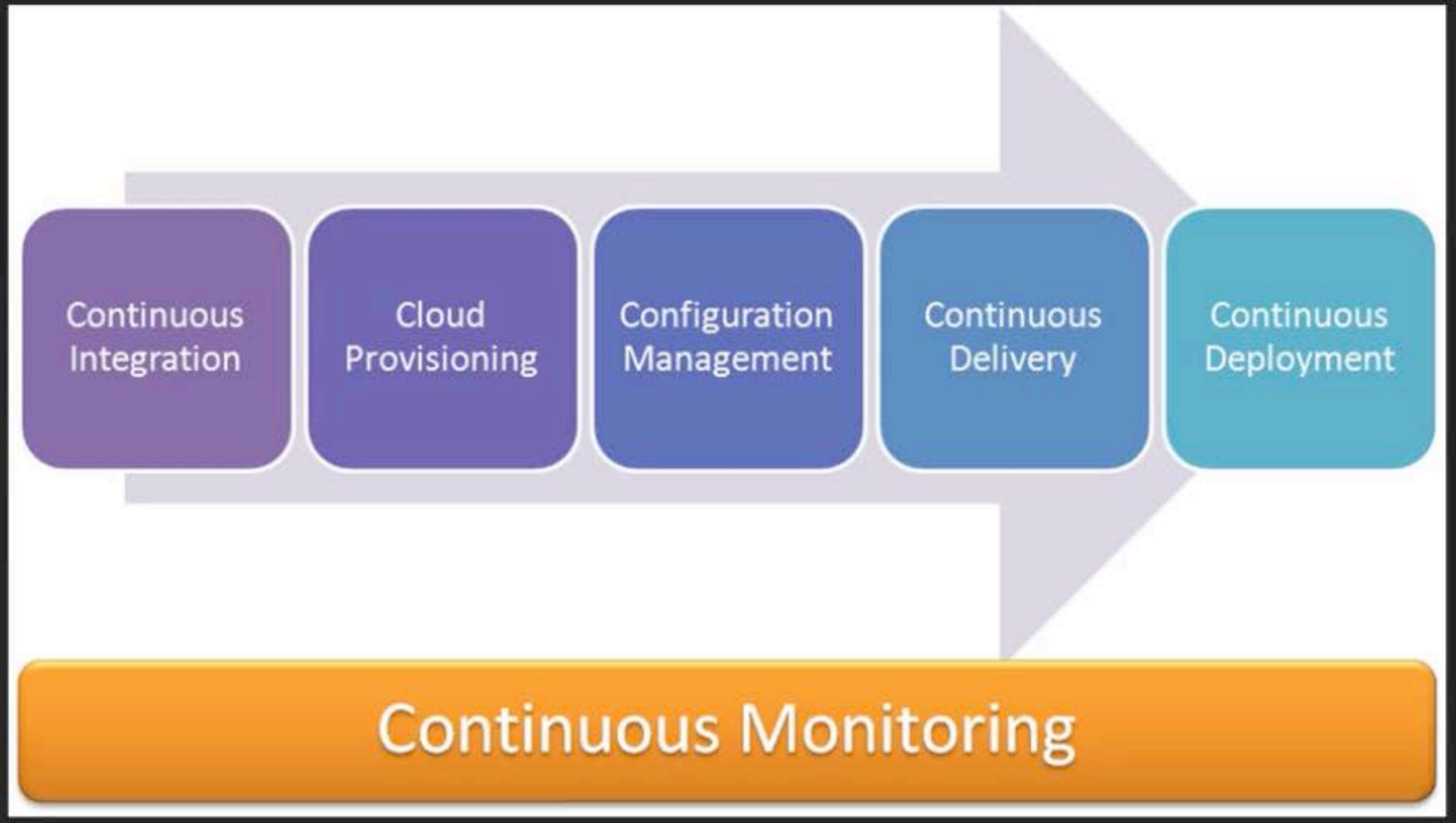
Steps to Use Microsoft Azure

- Create a SQL Server instance to host the database
- Import BACPAC files from the storage account to create a new database
- Deploy the web application to Microsoft Azure

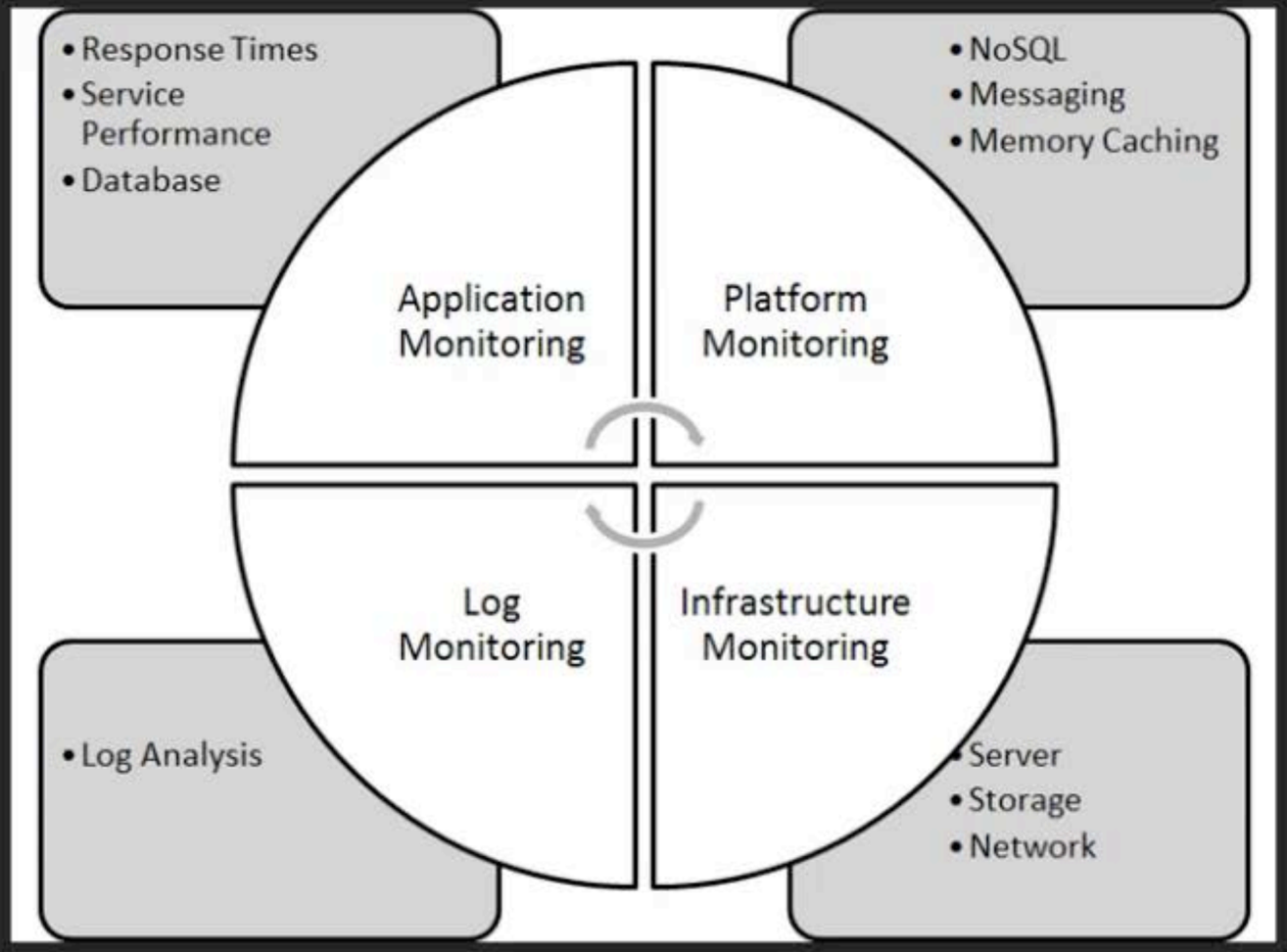
Best Practices for Continuous Delivery

- Repetitive tasks
- Difficult tasks
- Manual tasks

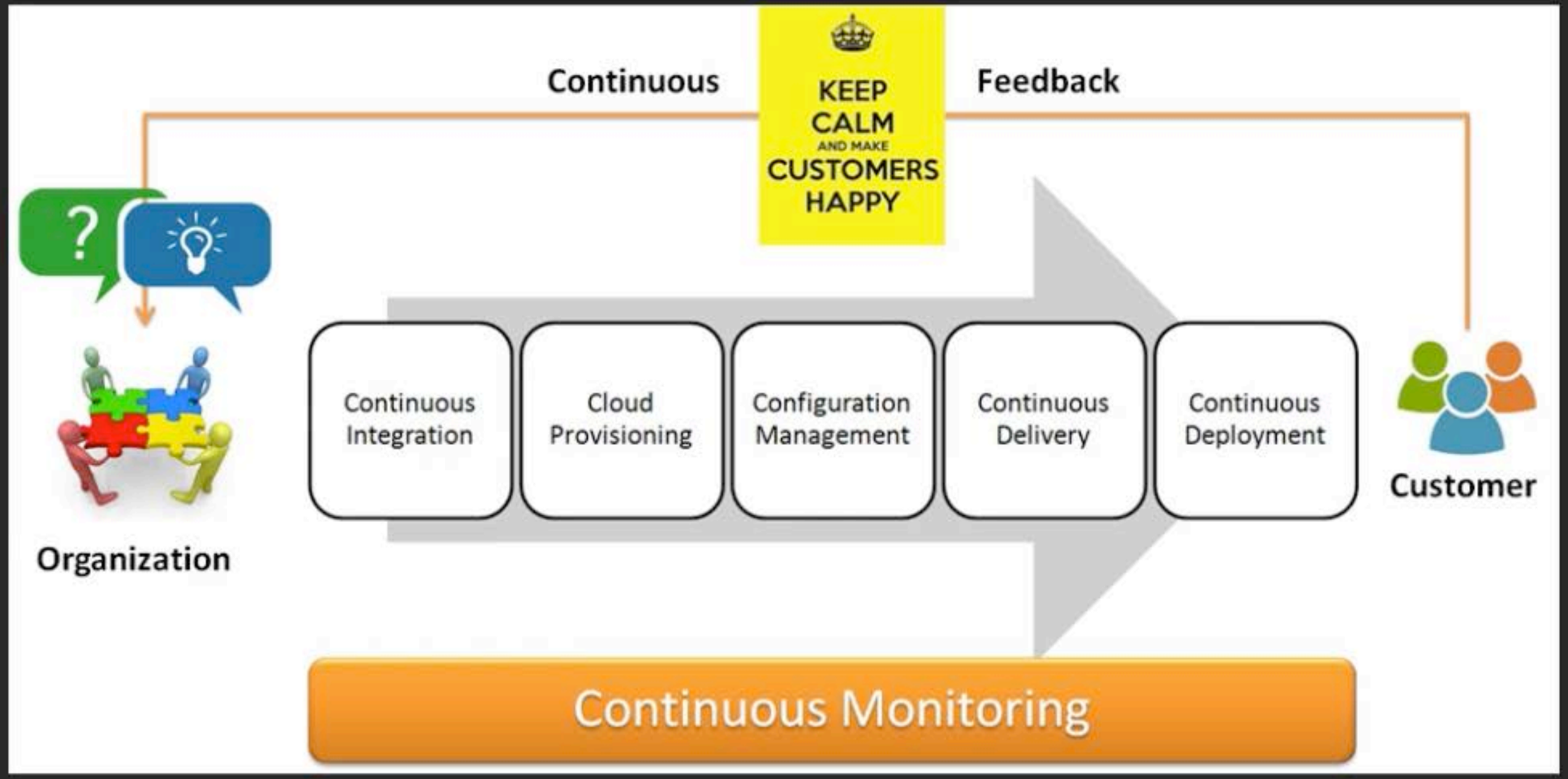
Continuous Monitoring



Types of Monitoring



Types of Monitoring



Next Video

Tools and Technologies