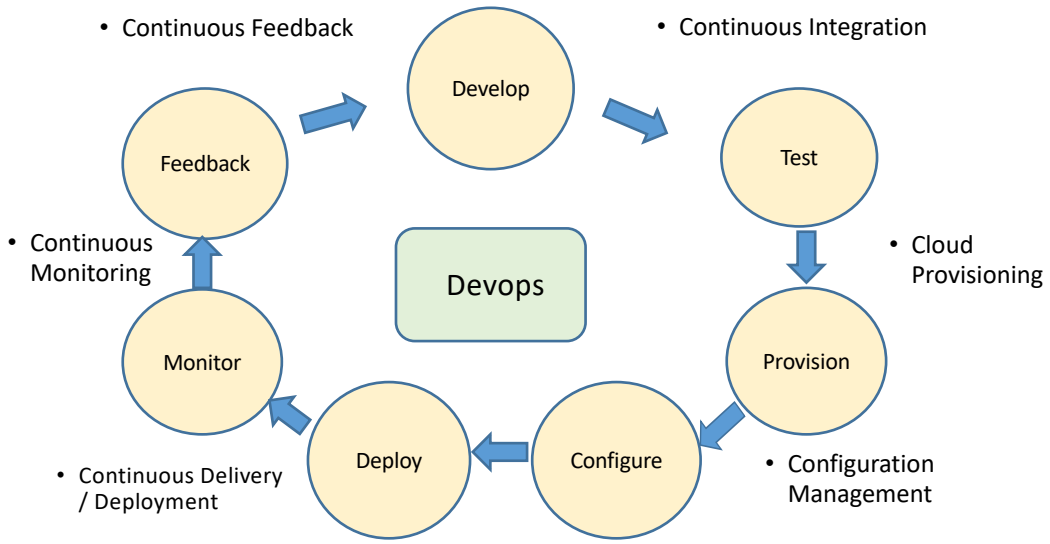
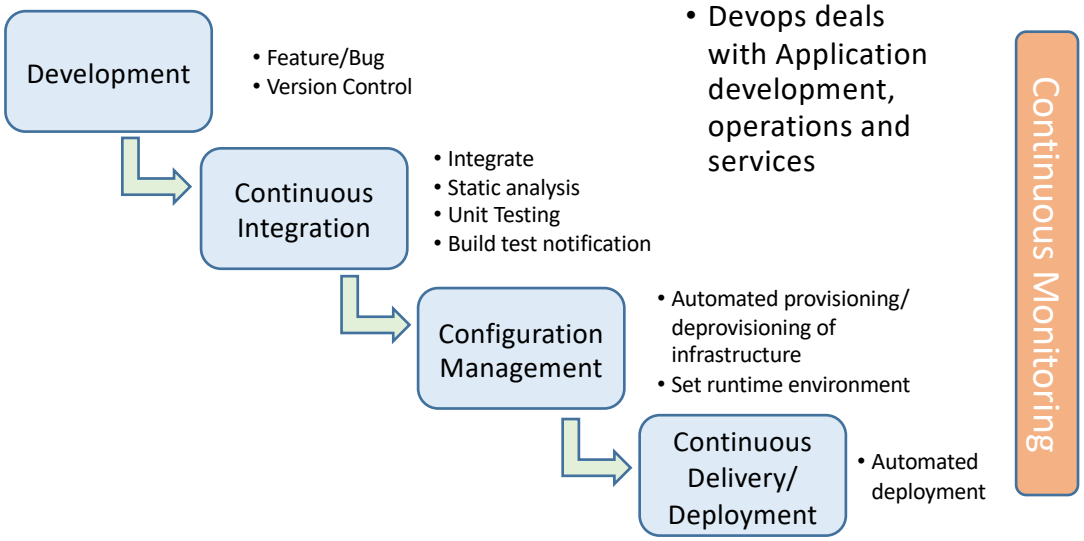
The Devops Lifecycle

Application Delivery Lifecycle



Mapping of an Application



Build Automation

- Compiling source code into class files
  - Or binary files
- Providing references to third party libraries
- Providing the path(s) of conifiguration files
- Packaging class files (or binary files)
  - Libraries
  - Jar files
  - War files

Build Automation

- Apache Ant
- Maven
- Sbt
- Gradle
- …

- Essential as the other components of the pipeline will depend on automated build/test capabilities

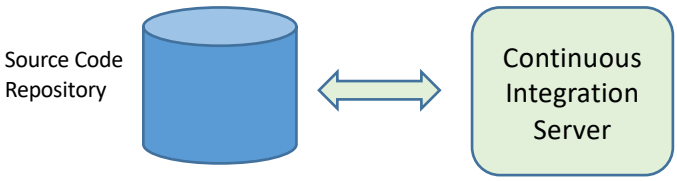
Continuous Integration

- Every check in of code by a developer is verified

- Pull mechanism
  - Execute automated build at specific times

- Push Mechanism
  - Execute automated build in reaction to code check in

Continuous Integration



- Automated build verification through integration of code from source code repository
- Unit test execution and static code analysis
- Notification management on build status
- Continuous feedback and deployment into environment as next step

Advantages of CI

- Helps identify bugs or errors in code at early stage of development
  - Makes it easier to cure and fix issues
- CI requires developers to integrate code into a shared repository several times daily
- Key component of the release management strategy of an organisation that wants to develop a Devops culture

Benefits of CI

- Automated integration
  - Pull or push
- Repeatable process not requiring any manual intervention
- Automated test case execution
- Coding standards verification

Benefits of CI (continued)

- Execution of scripts based on requirements
- Quick feedback
  - Build status notifications to stakeholders by, eg. Email
- Teams can focus on work and not managing process

Best Practices

- Maintain a code repository
  - Git
  - Subversion
- Check in 3rd party jar files, build scripts, other artifacts
- Execute builds fully from source code repository
  - Never underestimate the value of a clean build
- Automate
- Make the build self-testing

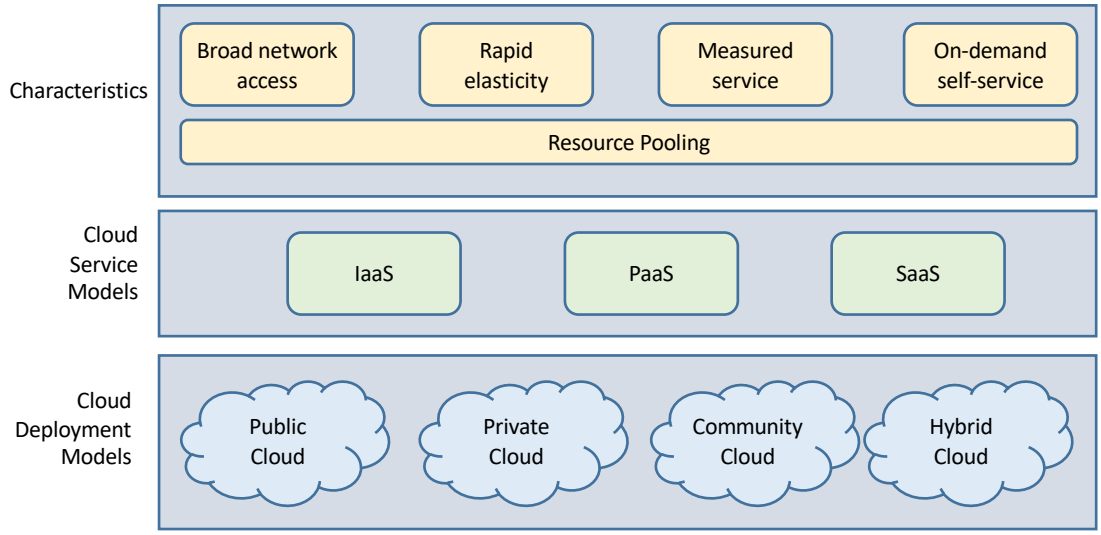
Best Practices

- Commit changes at least once per day per feature
- Ensure every commit is built to verify integrity of the changes
- Authenticate users and enforce access control
- Use alphanumeric characters for build names
  - Avoid symbols
  - Helps assign build execution to slave instances of the server, improving throughput

Best Practices

- Back up CI server's home directory regularly
  - Saves archived builds and other useful info
- Make sure CI server has enough disk space
- Don't run multiple build jobs concurrently
  - Unless using a master/slave setup
- Use e-mail, sms, twitter or other tool to notify project stakeholders of any issues

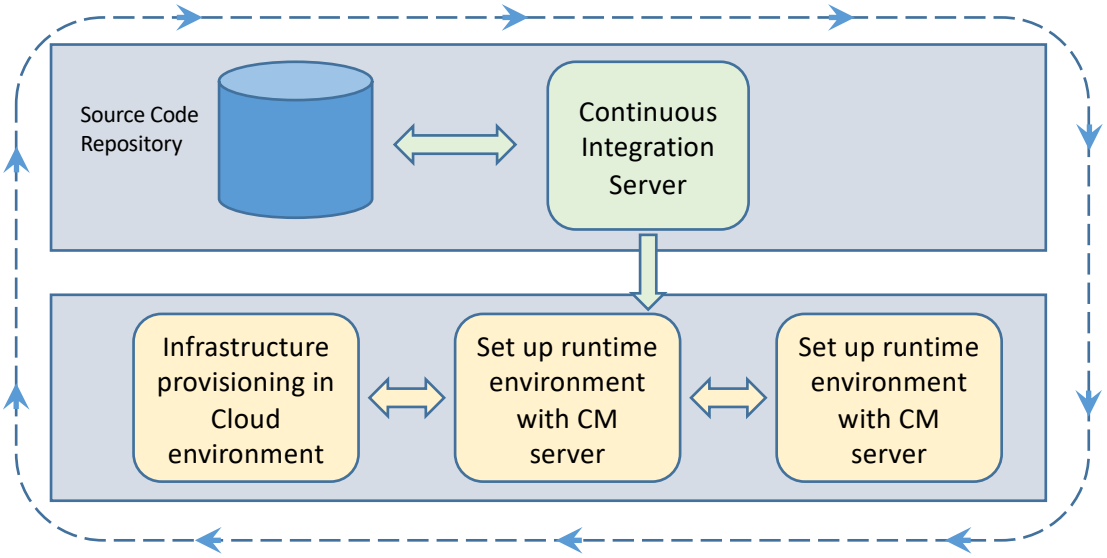
Deployment Models



Configuration Management

- Configuration Management (CM) keeps track of versions or details related to the state of specific nodes
  - Server runtime
- A centralised change can trigger this or nodes may communicate with the CM server
- CM tools make this process efficient when only change behaviour is to be updated
  - Eg changed network port numbers
- Entire installation and modification need not be applied again to server nodes
- Chef, Puppet, Ansible, Salt

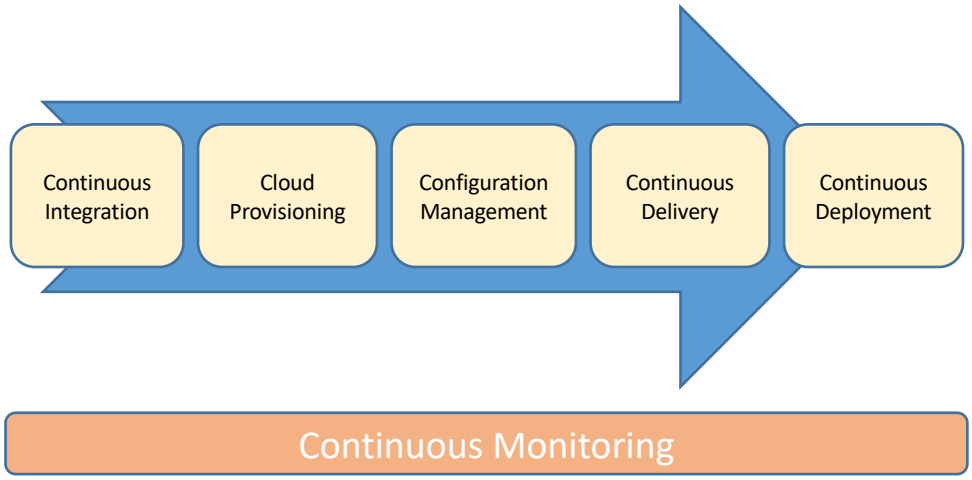
Continuous Delivery/Deployment



Best Practices

- Automate everything
- Especially
  - Repetitive tasks
  - Difficult tasks
  - Manual tasks

Continuous Monitoring



Monitoring and Feedback