Cloud Native

Cloud Native Scope

Cloud Native is structuring teams, culture and technology to utilize automation and architectures to manage complexity and unlock velocity.

Joe Beda, Heptio

Cloud Native Applications

Applications adopting the principles of microservices packaged as containers orchestrated by platforms running on top of cloud infrastructure.
A Kubernetes based Microservice
# Local vs Distributed Primitives

<table>
<thead>
<tr>
<th>Concern / Abstraction</th>
<th>Local / Java</th>
<th>Distributed / Kubernetes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behaviour encapsulation</td>
<td>Class, Object</td>
<td>Container Image, Container</td>
</tr>
<tr>
<td>Unit of reuse</td>
<td>.jar</td>
<td>Container Image</td>
</tr>
<tr>
<td>Deployment unit</td>
<td>.jar, .war, .ear</td>
<td>Pod</td>
</tr>
<tr>
<td>Buildtime/Runtime isolation</td>
<td>Module, Package, Class</td>
<td>Container Image, Namespace</td>
</tr>
<tr>
<td>Initialization preconditions</td>
<td>Constructor</td>
<td>Init-container</td>
</tr>
<tr>
<td>Post initialization</td>
<td>init-method</td>
<td>PostStart hook</td>
</tr>
<tr>
<td>Pre destroy</td>
<td>destroy-method</td>
<td>PreStop hook</td>
</tr>
<tr>
<td>Cleanup procedure</td>
<td>finalize(), ShutdownHook</td>
<td>Defer-container***</td>
</tr>
<tr>
<td>Asynchronous, Parallel execution</td>
<td>ThreadPoolExecutor, ForkJoinPool</td>
<td>Job</td>
</tr>
<tr>
<td>Periodic task</td>
<td>Timer, ScheduledExecutorService</td>
<td>CronJob</td>
</tr>
<tr>
<td>Background task</td>
<td>Daemon Thread</td>
<td>DaemonSet</td>
</tr>
<tr>
<td>Configuration management</td>
<td>Properties</td>
<td>ConfigMap, Secret</td>
</tr>
<tr>
<td>Service discovery</td>
<td>ZooKeeper, Consul...</td>
<td>Service</td>
</tr>
</tbody>
</table>
Principles of Software Design

- KISS - Keep it simple, stupid
- DRY - Don't repeat yourself
- YAGNI - You aren't gonna need it
- SoC - Separation of concerns
- SOLID Principles by Robert C. Martin:
  - Single responsibility
  - Open/Closed
  - Liskov substitution
  - Interface segregation
  - Dependency inversion
Principles for Cloud Native Applications

- **Build time:**
  - Single Concern Principle (SCP)
  - Self-Containment Principle (S-CP)
  - Image Immutability Principle (IIP)

- **Runtime:**
  - High Observability Principle (HOP)
  - Lifecycle Conformance Principle (LCP)
  - Process Disposability Principle (PDP)
  - Runtime Confinement Principle (RCP)

Single Concern Principle

Design patterns:
- Sidecar
- Ambassador
- Adapter
- Init-container
- Defer-container
Self-Containment Principle

Anti-patterns:
- Locomotive
Image Immutability Principle

Also known as:
- Dev/Prod parity
- Impedance mismatch
- Snowflakes vs Phoenix
High Observability Principle

Java implementations:
- Spring Boot Actuator
- Dropwizard Metrics
- WildFly Swarm Monitor
- MicroProfile Healthchecks
Lifecycle Conformance Principle

- Graceful shutdown:
  - SIGTERM
  - SIGKILL

- Lifecycle hooks:
  - PreStop
  - PostStart
Process Disposability Principle

Also known as:

- Cattle rather than pets
  - Don’t rely on a particular instance.
  - Be aware of shots at your cattle.
  - Be robust against sudden death.
- Fast startup
- Graceful shutdown
- Stateless service
Runtime Confinement Principle

Implications:
- Pod scheduling
- Pod auto scaling
- Pod eviction
- Pod QoS classes:
  - Guaranteed
  - Burstable
  - Best Effort
- Cluster capacity management
Principles, Patterns...

Cloud Native for Developers

Primitives & Abstractions

Design Principles

Techniques & Practices

Design Patterns
Techniques and Practices

- **Aim for small images** - this reduces container size, improves build, and deployment time.
- **Support arbitrary user IDs** - avoid using the sudo command or requiring a specific user ID.
- **Mark important ports** - declare ports using the EXPOSE command.
- **Use volumes for persistent data** - the data that needs to be preserved after a container is destroyed.
- **Set image metadata** - Image metadata in the form of tags, labels, and annotations.
- **Synchronize host and image** - attributes such as time and machine ID.
- **Log to STDOUT and STDERR** - to ensure container logs are picked up and aggregated properly.
Kubernetes Resources

- Principles of container-based application design (white paper)

- Design patterns for container-based distributed systems (white paper)

- Kubernetes Patterns (ebook)
  https://leanpub.com/k8spatterns

- Designing Distributed Systems (ebook)
  http://shop.oreilly.com/product/0636920072768.do

- Kubernetes in Action (ebook)
  https://www.manning.com/books/kubernetes-in-action
THANK YOU

plus.google.com/+RedHat
linkedin.com/company/red-hat
facebook.com/redhatinc
twitter.com/RedHat
youtube.com/user/RedHatVideos